ITANIUM® SOLUTIONS ALLIANCE

SUCCESS IS YOURS FOR THE TAKING. JOIN US.

Itanium® Extensible Firmware Interface (EFI)

Purpose of this session

1. Explain the basic design and implementation of the EFI (Extensible Firmware Interface)

2. Show simple examples of EFI shell usage

3. Show the usual usage of EFI menus

Extensible Firmware Interface (EFI) Module Topics

- Acronyms
- Extensible Firmware Interface (EFI) design
- EFI Implementation
- EFI User Interface

Some Acronyms

- ACPI - Advanced Configuration and Power Interface
- EFI - Extensible Firmware Interface
- FIT - Firmware Interface Table
- ICMB - Intelligent Chassis Management Bus
- IPMI - Intelligent Platform Management Interface
- IPMB - Intelligent Platform Management Bus
- PAL - Processor Abstraction Layer
- SAL - System Abstraction Layer

EFI Design

EFI - Design Goals

- Real mode BIOS INT based solution was not an option
  - A chance to move beyond limitations of the PC BIOS

- *The idea was to get as much platform independence as possible*

- EFI allows:
  - Abstract and replace legacy devices, add new boot devices
  - High level language development (drivers to be written in C)
    - A standardized driver model
    - Common source code for IA-32 and Itanium®

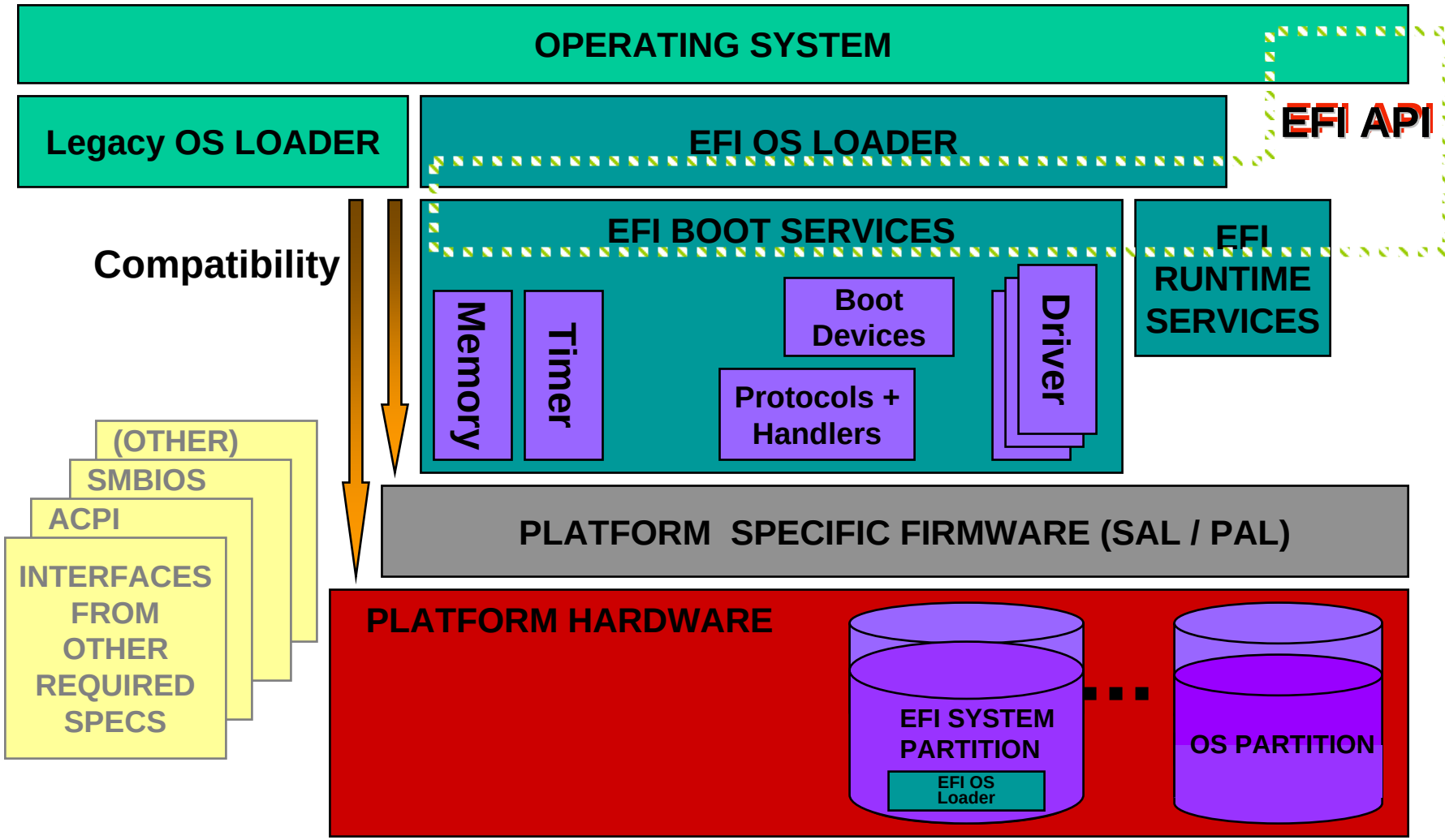
  - Platform value-add
    - Platform management utilities outside OS
    - A Pre-OS point in system startup and administration, portable OS neutral tools
    - Bootable flash update CD without DOS

EFI - Design Features

- OS neutral - can be used by multiple OSes
- Focus is on the API between OS/boot loader, and the firmware
  - Not on the User Interface or utilities
  - EFI shell is a reference implementation of a user interface; vendors may improve it
  - Available utilities and tools change with time and manufacturer
- Intended to be implemented and extended (in a competitive manner) by system manufacturers
- Shell and Boot Manager are main user interactions
- ACPI (Advanced Configuration and Power Interface) based

EFI Implementation

EFI in the Firmware Stack



EFI Services

- Runtime Services
  - Abstract minor parts of the hardware implementation from the OS
- Boot services interfaces
  - Global boot service interfaces
  - Device handle-based boot service interfaces
  - Device protocols
  - Protocol services

EFI Disc Format – GUID Partition Table (GPT)

- Logical Block Addressing is 64 bits and supports many partitions
- Uses a primary and backup table for redundancy and version number and size fields for future expansion
- Uses CRC32 fields for improved data integrity
- Uniquely identifies each partition
- Uses a GUID and attributes to define partition content type
- Each partition contains a 36 Unicode character human readable name.

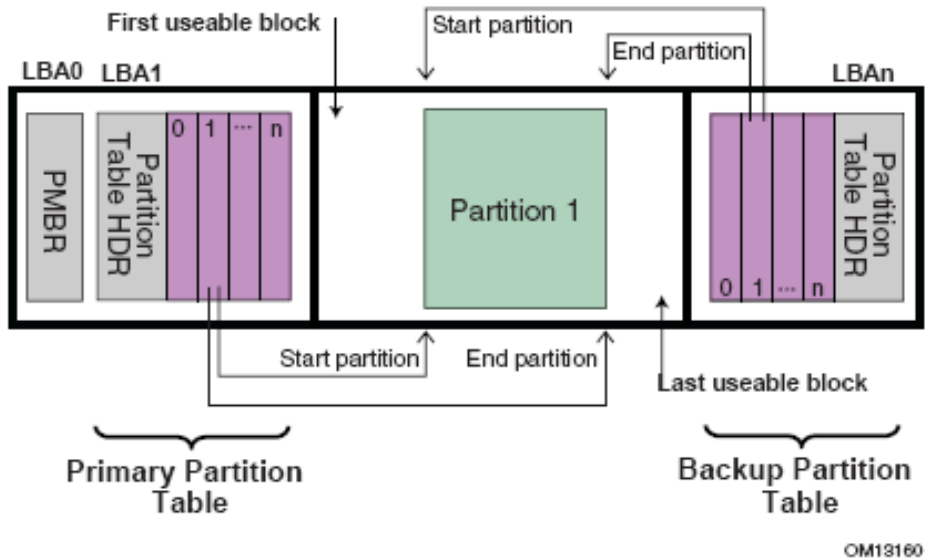


Figure 11-2. GUID Partition Table (GPT) Scheme

EFI Disc Format – GPT (cont.)

Partition Discovery

1. Check for GUID Partition Table Headers
2. Follow ISO-9660 specification to search for ISO-9660 volume structures on the magic LBA
   - "El Torito" CDROM spec
3. If none of the above, check LBA 0 for a legacy MBR partition table
4. No partition found on device

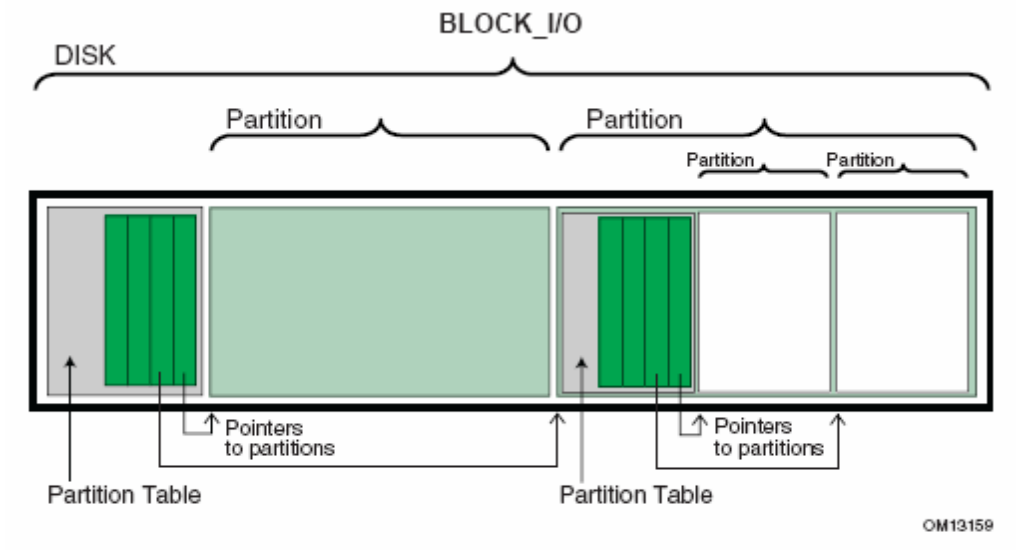


Figure 11-1. Nesting of Legacy MBR Partition Records

EFI – Low level firmware
Processor Abstraction Layer (PAL)

- Abstracts processor specific events
  - reset, init, MCA into a standard interface
- Separates supporting hardware from the processor(s)
- Simplifies processor upgrade and interchange
  - Provides a consistent interface as processors change
- The benefits are:
  - Faster implementation of new family members (i.e: Madison, Montecito)
  - Investment protection through cost reduction

EFI – Low level firmware
System Abstraction Layer (SAL)

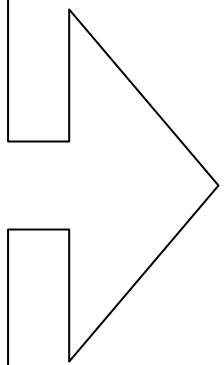
- Shields OS and other higher level software from implementation differences in the platform

- It provides:
  - Initialization, configuration, and testing of the hardware platform
  - Initialization and configuration info to OS and monitoring system
  - Integration of the processor(s) with the rest of the system (through PAL)
  - The environment for EFI and the OS loader
  - Runtime services to the OS and EFI

EFI Environment for Operating Systems

Before EFI

- 16bit Real Mode
- Legacy BIOS calls
- E820 Memory Map
- VGA
- Lilo, grub, etc
- Statically reserved memory regions
- Old Partition Setup



EFI Environment

- Native Mode
- EFI services
- UGA
- EFI boot adapter
- No special memory region reservation
- EFI system Partition

EFI Resources

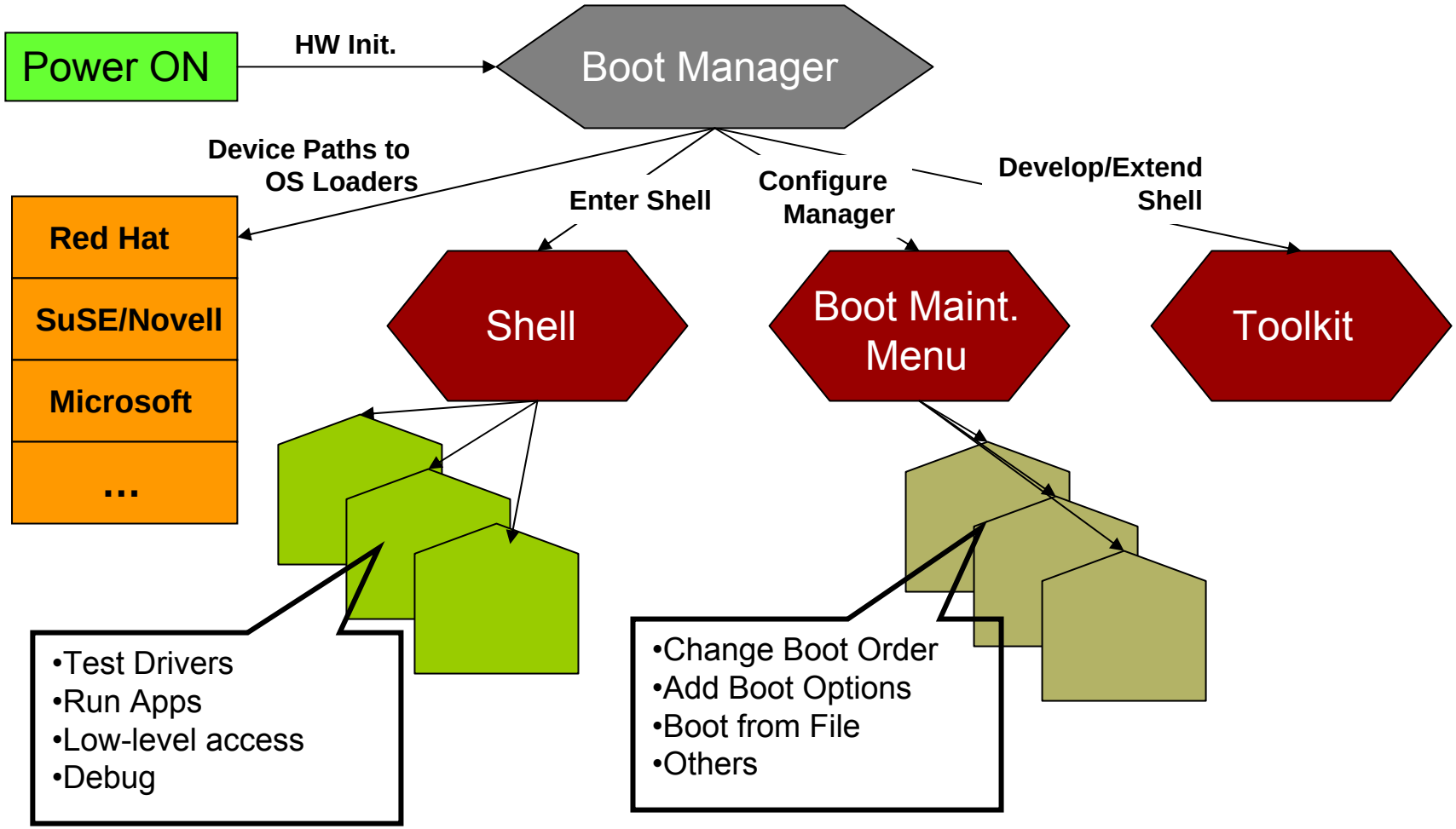
- EFI website at Intel
  - http://www.intel.com/technology/efi/efi.htm
- Previous EFI related presentations
  - http://www.intel.com/technology/efi/efi.htm
- EFI Toolkit
  - http://www.intel.com/technology/efi/toolkit_overview.htm
  - Some of the content:
    - All source and makefiles to build IA-32 and Itanium versions of the toolkit
    - Utilities (hex dump of disk or media)
    - Standard C library
    - Network stack and network utilities
    - PPP network support
    - Python interpreter
    - Text editor

EFI User Interface

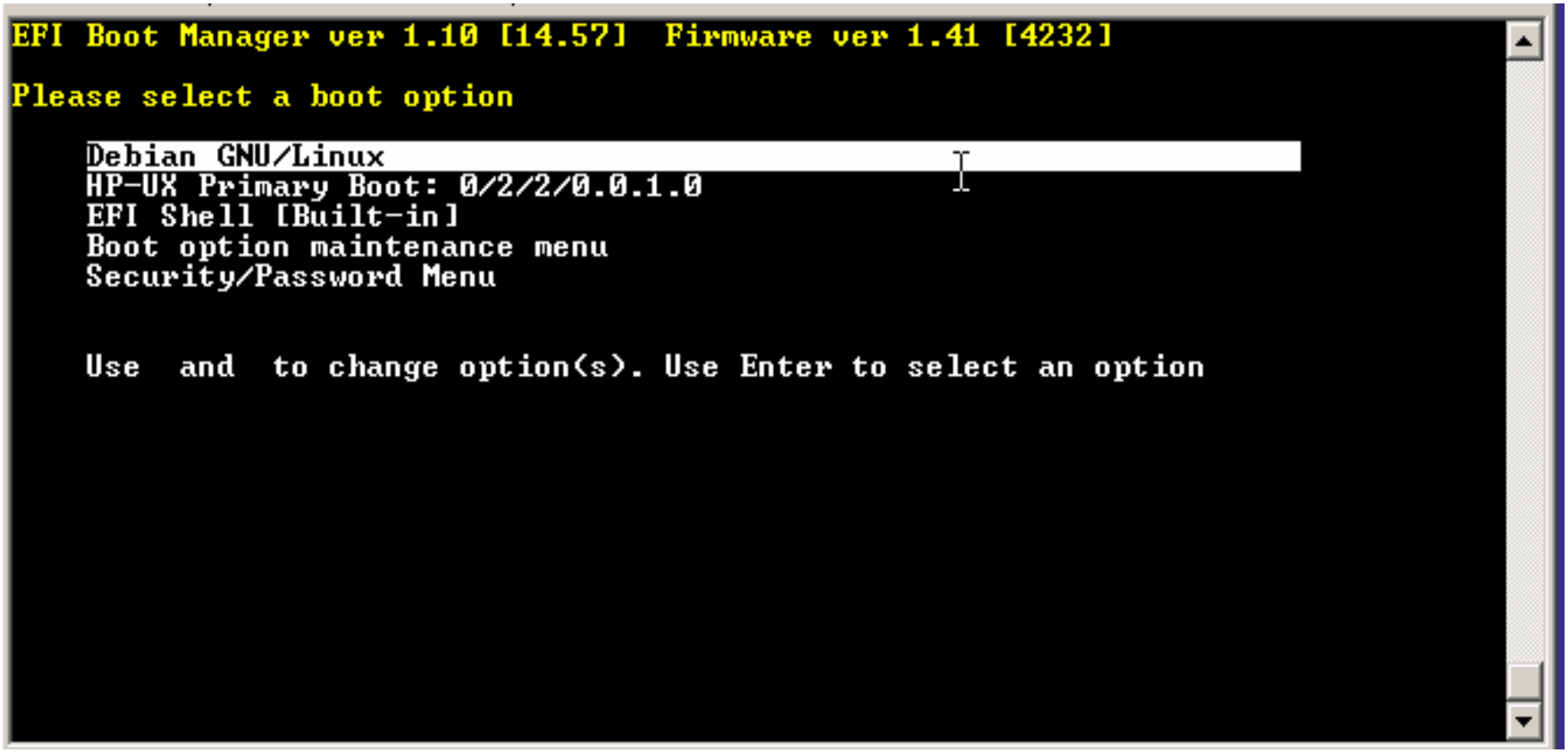
Section Topics

- The Flow of EFI
- EFI Main menu
- EFI Shell
- EFI Boot Manager
  - Boot entries
  - Console entries

The Flow of EFI

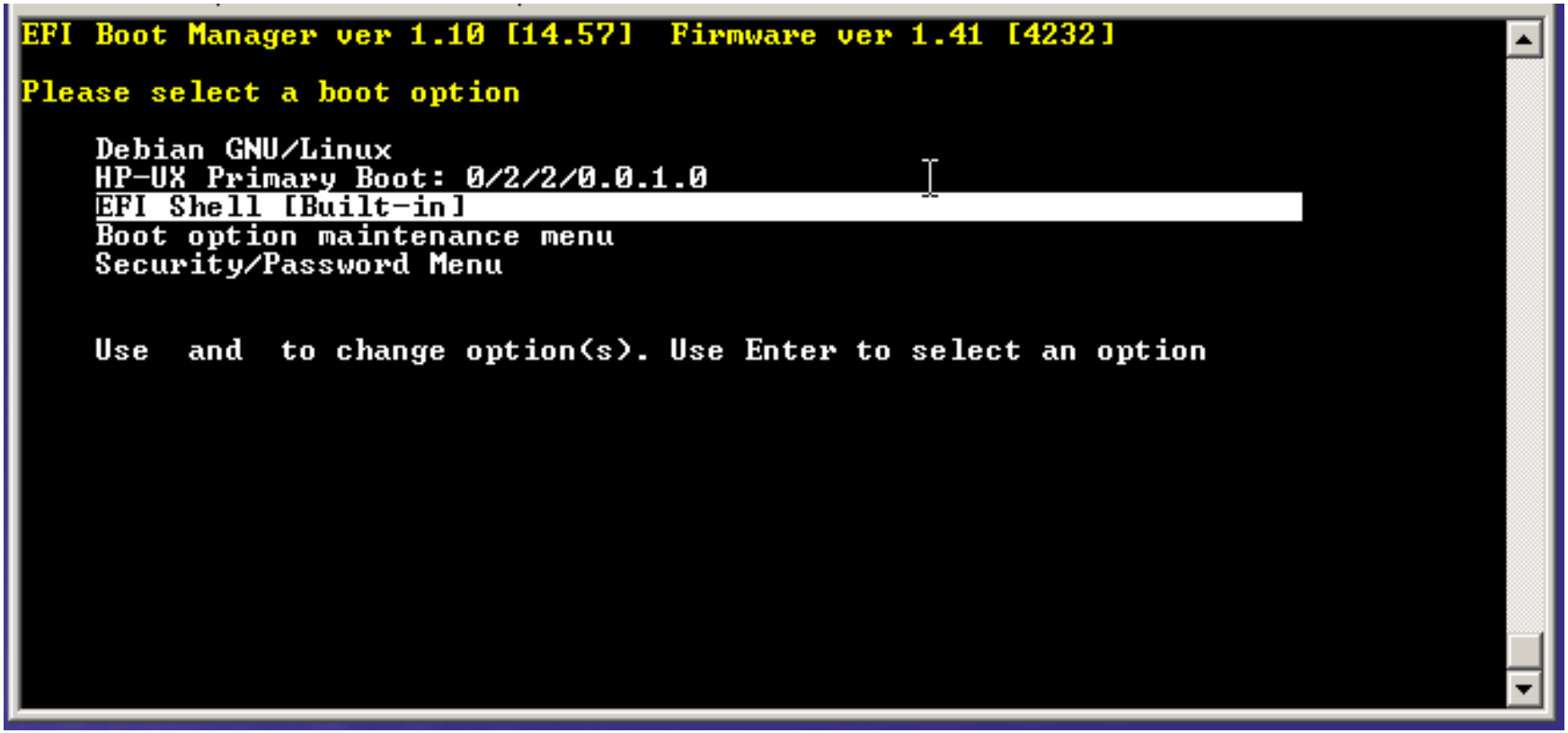


EFI – Main menu



The up and down arrows are used to move between choices

EFI – Selecting the EFI Shell



To invoke the EFI shell, move the highlight to its Menu entry, and hit Enter

EFI Shell

	EFI
Boot OS	√
C Language Implementation	√
Test Driver	√
Load Protocol	√
Text Editing	√
Networking	√
Launch custom Apps	√
Write Scripts	√
HW User Access	√
View/Manipulate Memory	√

EFI – Interactive use – EFI Shell

When the EFI shell is invoked:

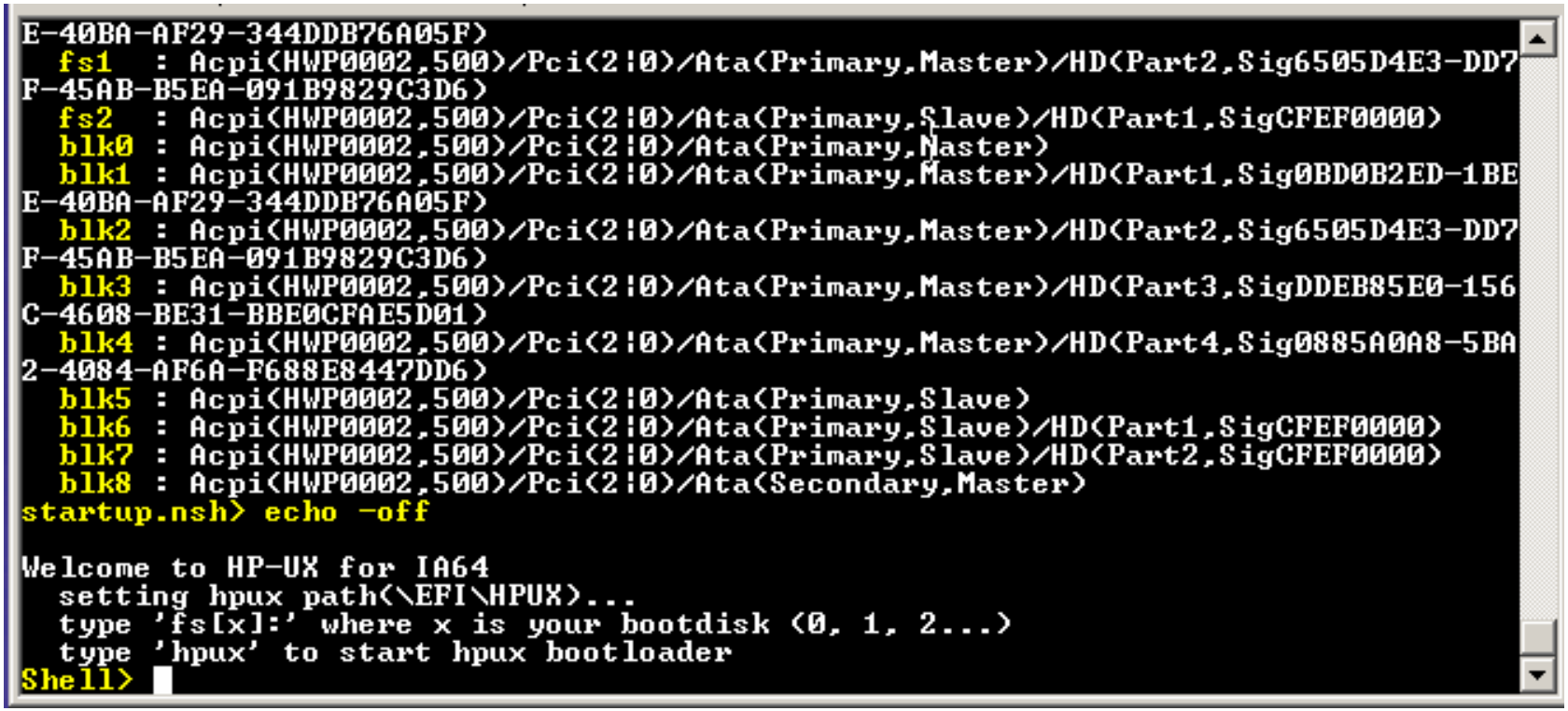
- Executes commands in the ‘startup.nsh’ file if it exists in the execution path
- Searches all partitions on the different media for EFI file systems and assigns mappings (fs0:, fs1:, fs2:…)
  - A disc with no EFI (FAT) partition will not have an *fsx:* mapping assigned, but it will have a raw device (*blkx:)* assigned to it
- Displays the mappings found, and issues the ‘Shell>’ prompt.
- No drive is assigned as the current drive by default.

EFI Shell - startup - ideal

```
EFI Boot Manager ver 1.10 [14.56]  Firmware ver 80.10 [4216]

Please select a boot option

    EFI Shell [Built-in]
    Boot option maintenance menu
    Security/Password Menu


    Use    and    to change option(s). Use Enter to select an option
Loading.: EFI Shell [Built-in]
POSSE Library version  0. 9 is loading...
    CellularPlatform = FALSE (use "setcell" to toggle)
EFI Shell version 1.10 [14.56]
Device mapping table
  fs0  : Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun0,Lun0)/HD(Part1,SigFA8B0000)
  blk0 : Acpi(HWP0002,0)/Pci(2|0)/Ata(Primary,Master)
  blk1 : Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun0,Lun0)
  blk2 : Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun0,Lun0)/HD(Part1,SigFA8B0000)
  blk3 : Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun0,Lun0)/HD(Part2,SigFA8B0000)
startup.nsh> echo -off
Welcome to HP-UX for IA64
  setting hpux path(\EFI\HPUX)...
  type 'fs[x]:' where x is your bootdisk (0, 1, 2...)
  type 'hpux' to start hpux bootloader
Shell>
```

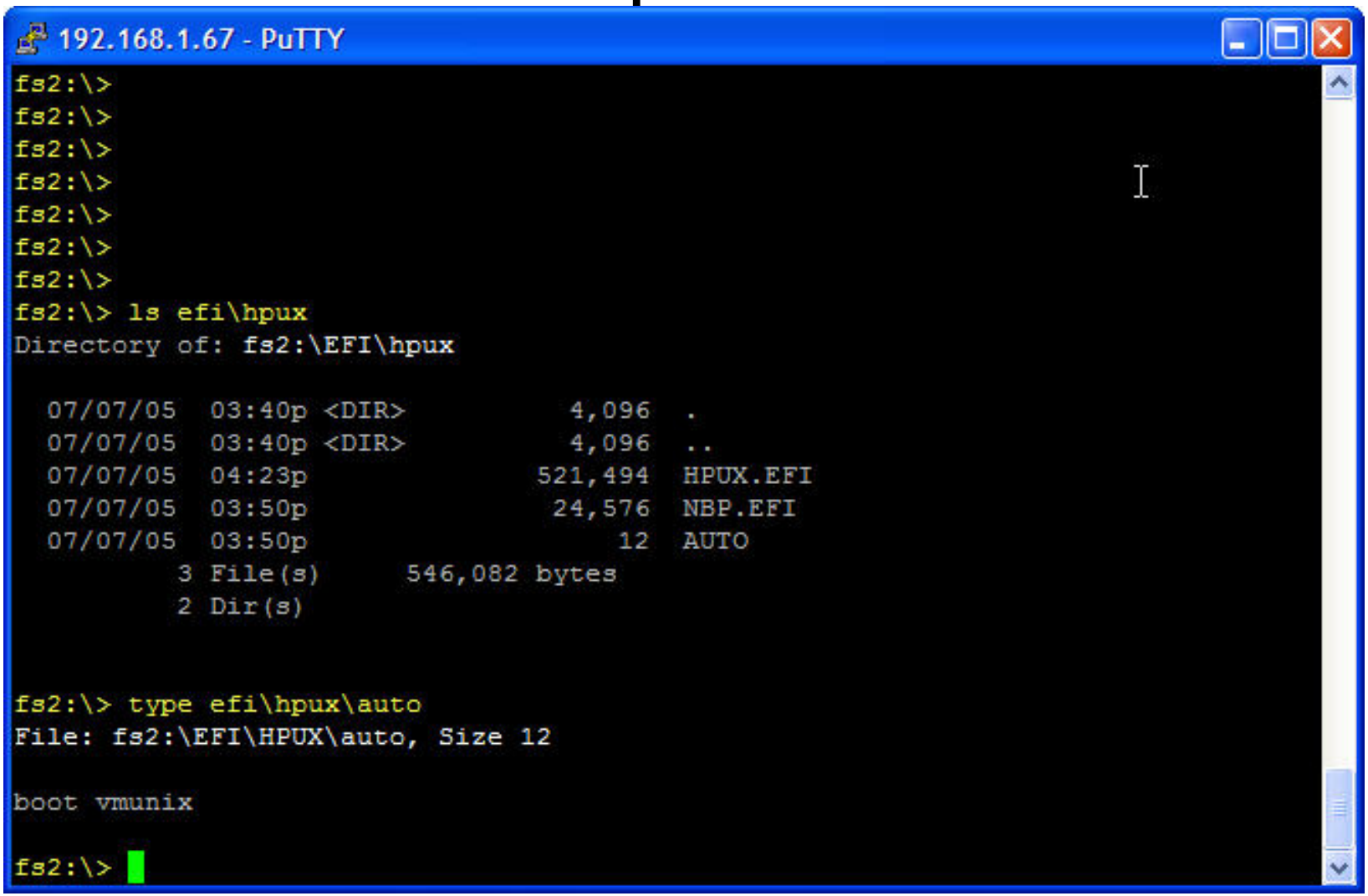
Here, we can see all the available disc devices, and decide which one we want to work with

EFI Shell startup – more likely

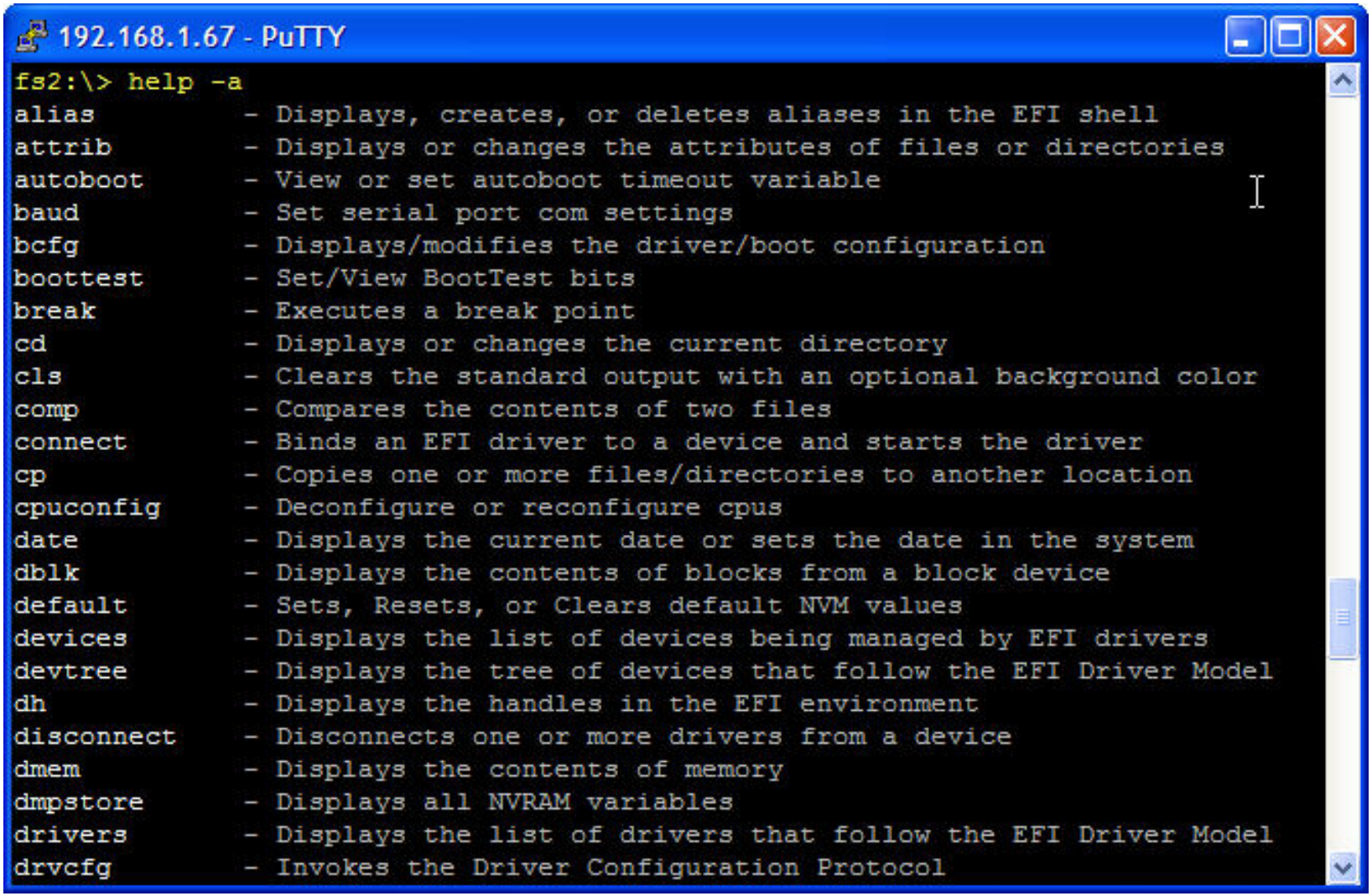


Here, we don't know what device is associated with fs0: - have to guess or figure it out. On a terminal emulator, we can probably scroll back up to see.

EFI – Shell command example



EFI – Shell command example

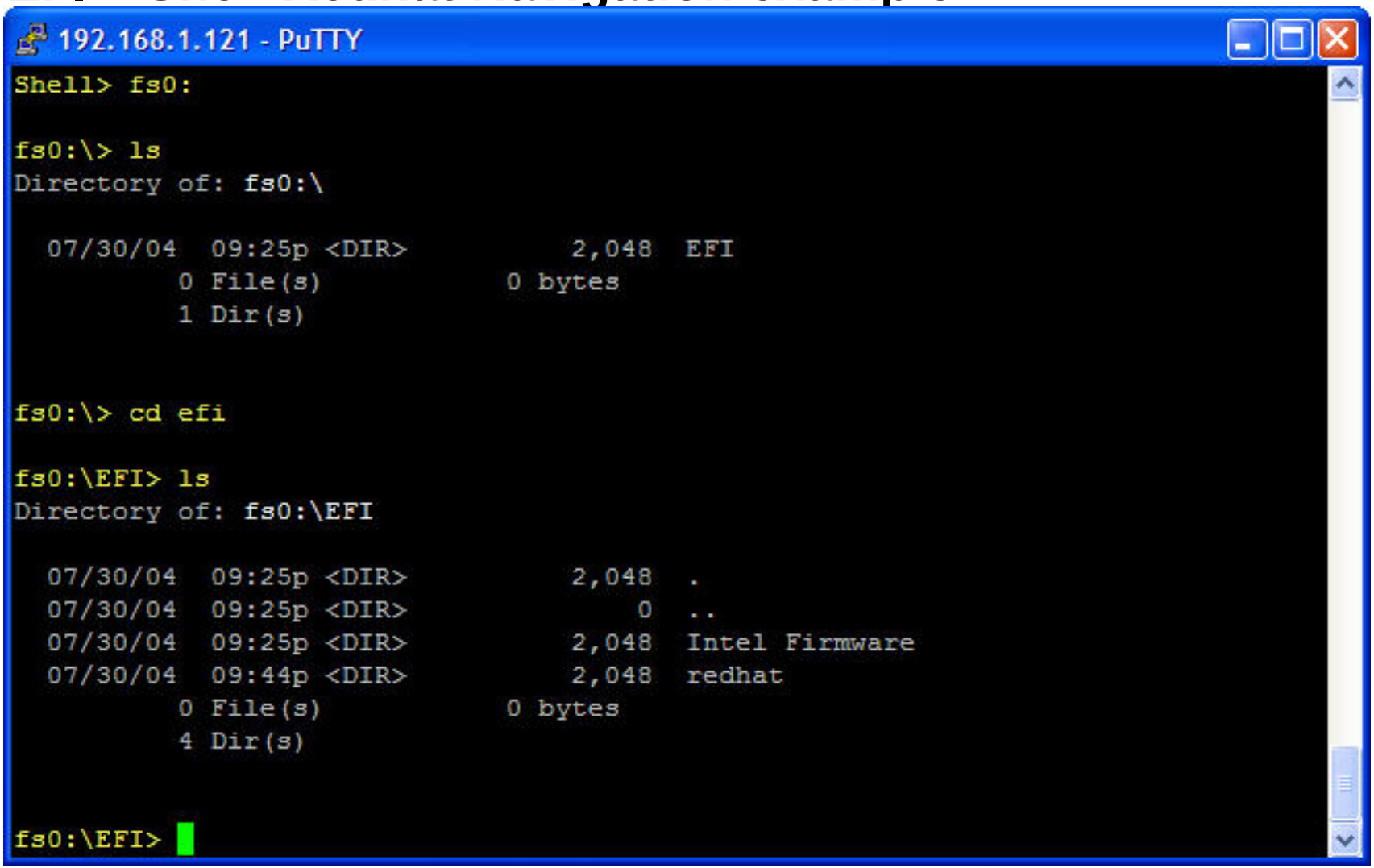


EFI – Shell syntax comparison

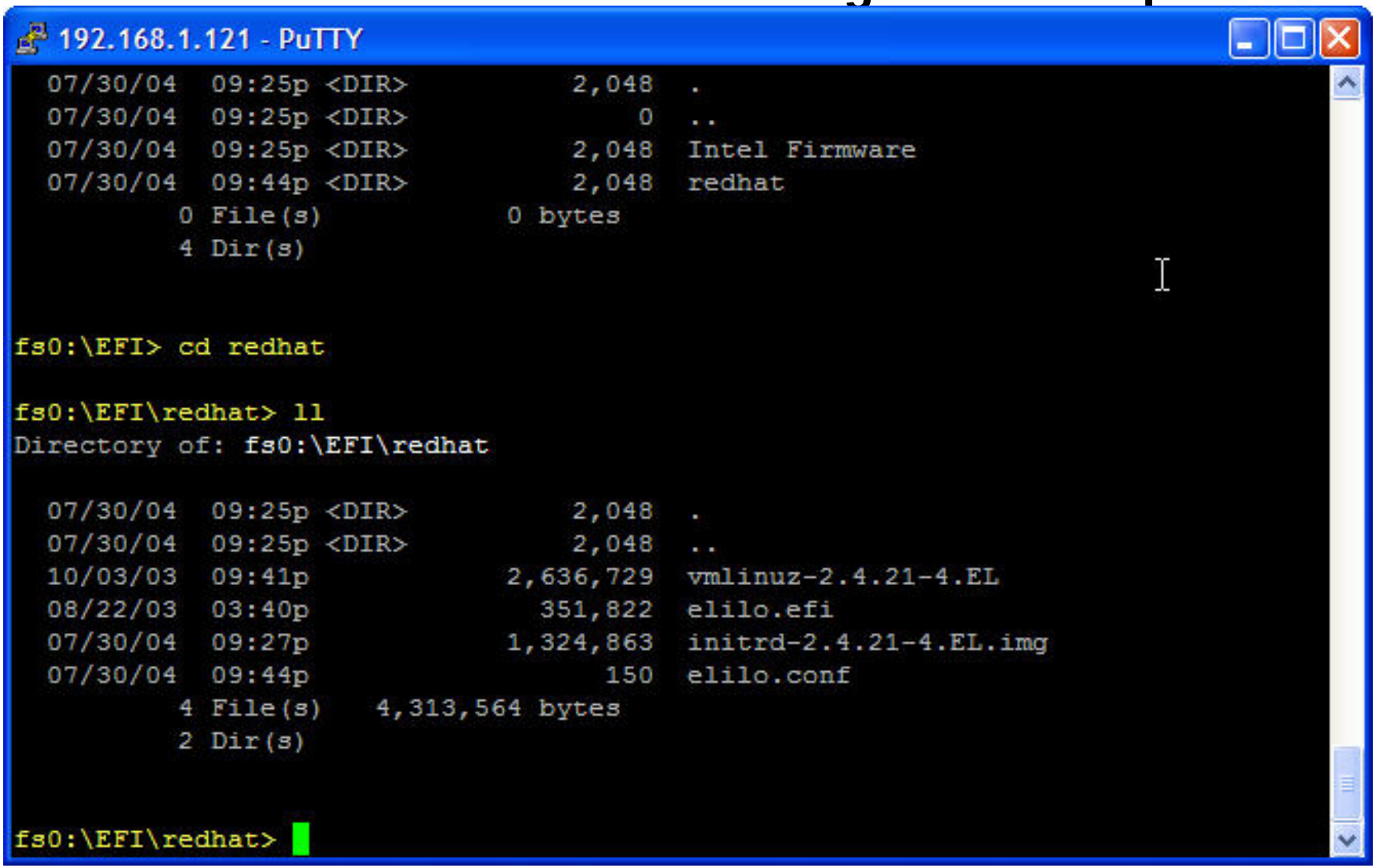
EFI	bash	DOS
\	/	\
fs0:	/	c:
ls	ls	dir
attrib	chmod	attrib
cp	cp	copy
rm	rm	del
mv	mv	move
mkdir	mkdir	md
%var%	$var	%var%
.efi, .nsh	anything, .sh	.exe, .bat,cmd
comp	diff/comm	fc

Redhat EFI shell example

EFI – Shell Redhat Navigation example



EFI – Shell command Redhat Navigation example



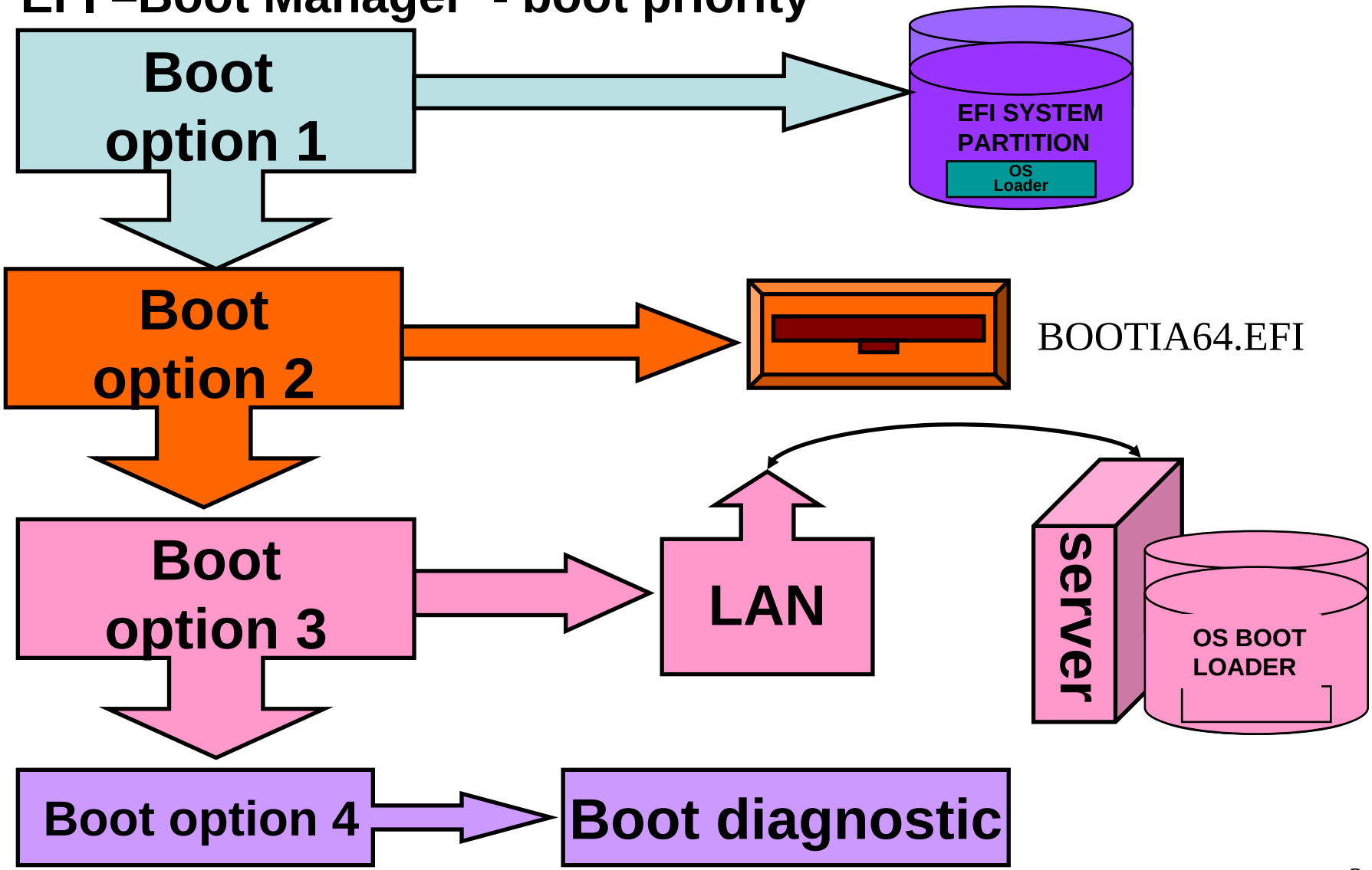
EFI – Shell command Redhat Navigation example

192.168.1.121 - PuTTY

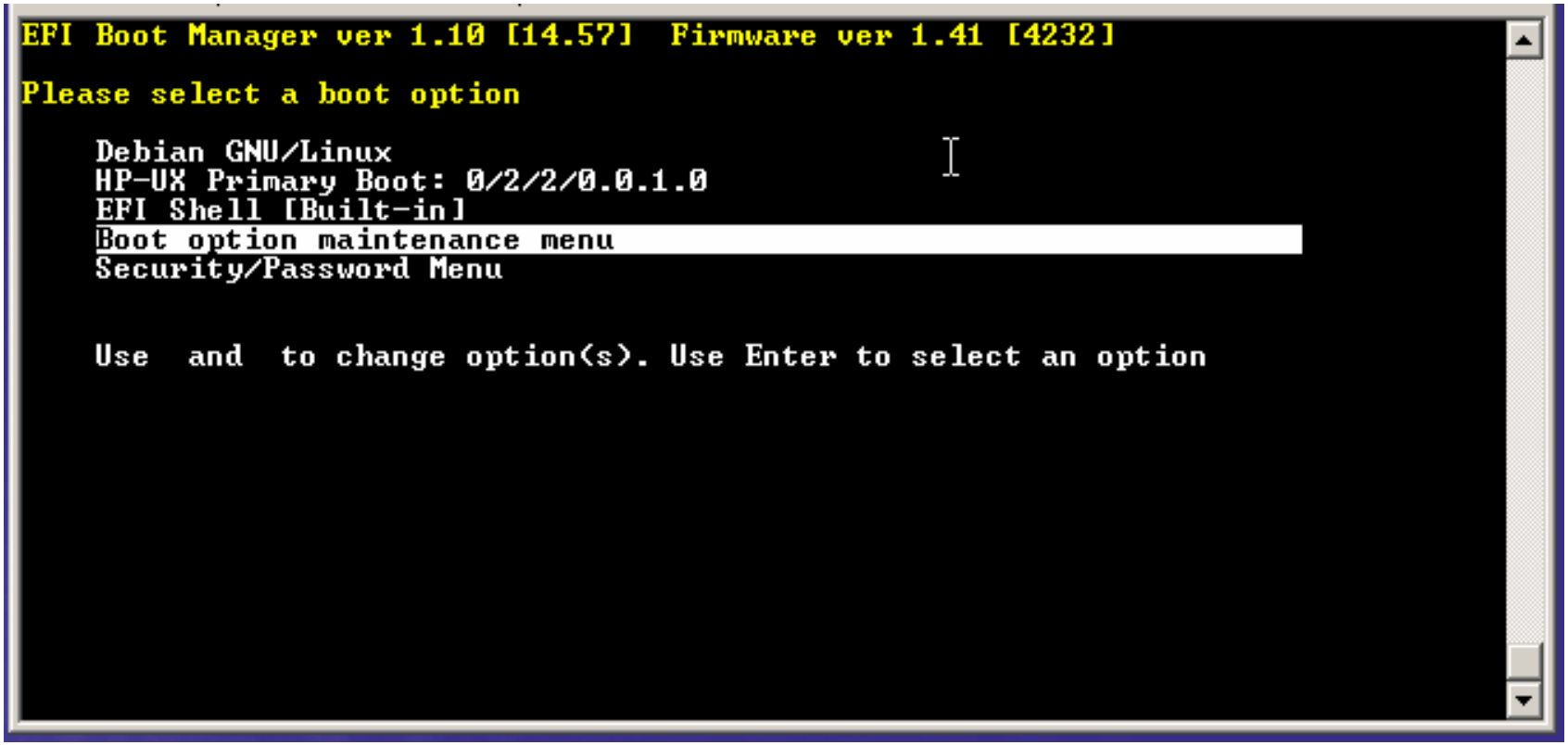
```
 07/30/04  09:25p <DIR>          2,048  .
 07/30/04  09:25p <DIR>          2,048  ..
 10/03/03  09:41p             2,636,729  vmlinuz-2.4.21-4.EL
 08/22/03  03:40p               351,822  elilo.efi
 07/30/04  09:27p             1,324,863  initrd-2.4.21-4.EL.img
 07/30/04  09:44p                   150  elilo.conf
         4 File(s)    4,313,564 bytes
         2 Dir(s)

fs0:\EFI\redhat> cat elilo.conf
File: fs0:\EFI\redhat\elilo.conf, Size 150

prompt
timeout=50
default=linux

image=vmlinuz-2.4.21-4.EL
        label=linux
        initrd=initrd-2.4.21-4.EL.img
        read-only
        append="hda=ide-scsi root=LABEL=/"

fs0:\EFI\redhat>
```

EFI Boot Manager Menus

EFI –Boot Manager - boot priority

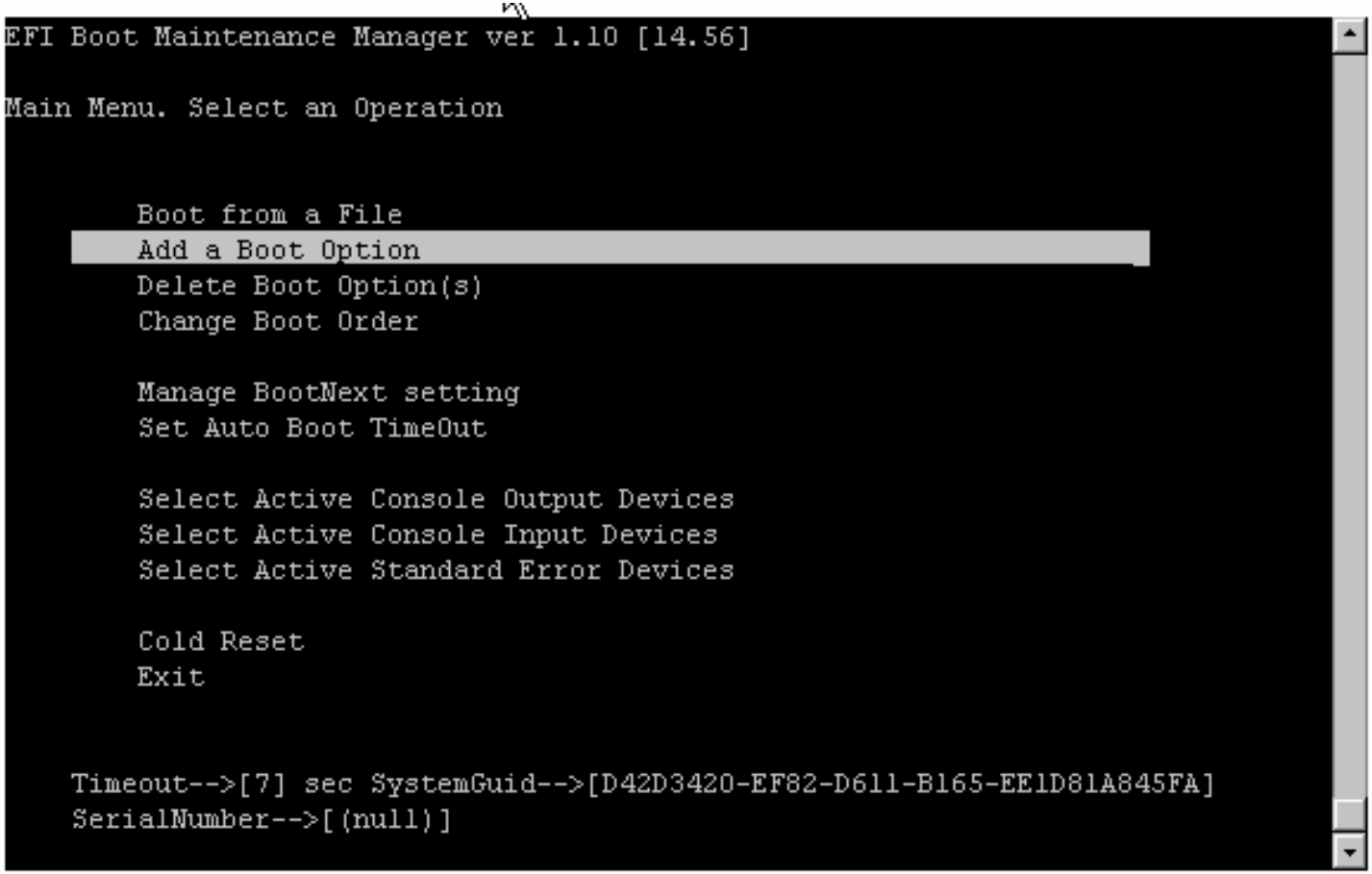


EFI Shell – Setting boot options



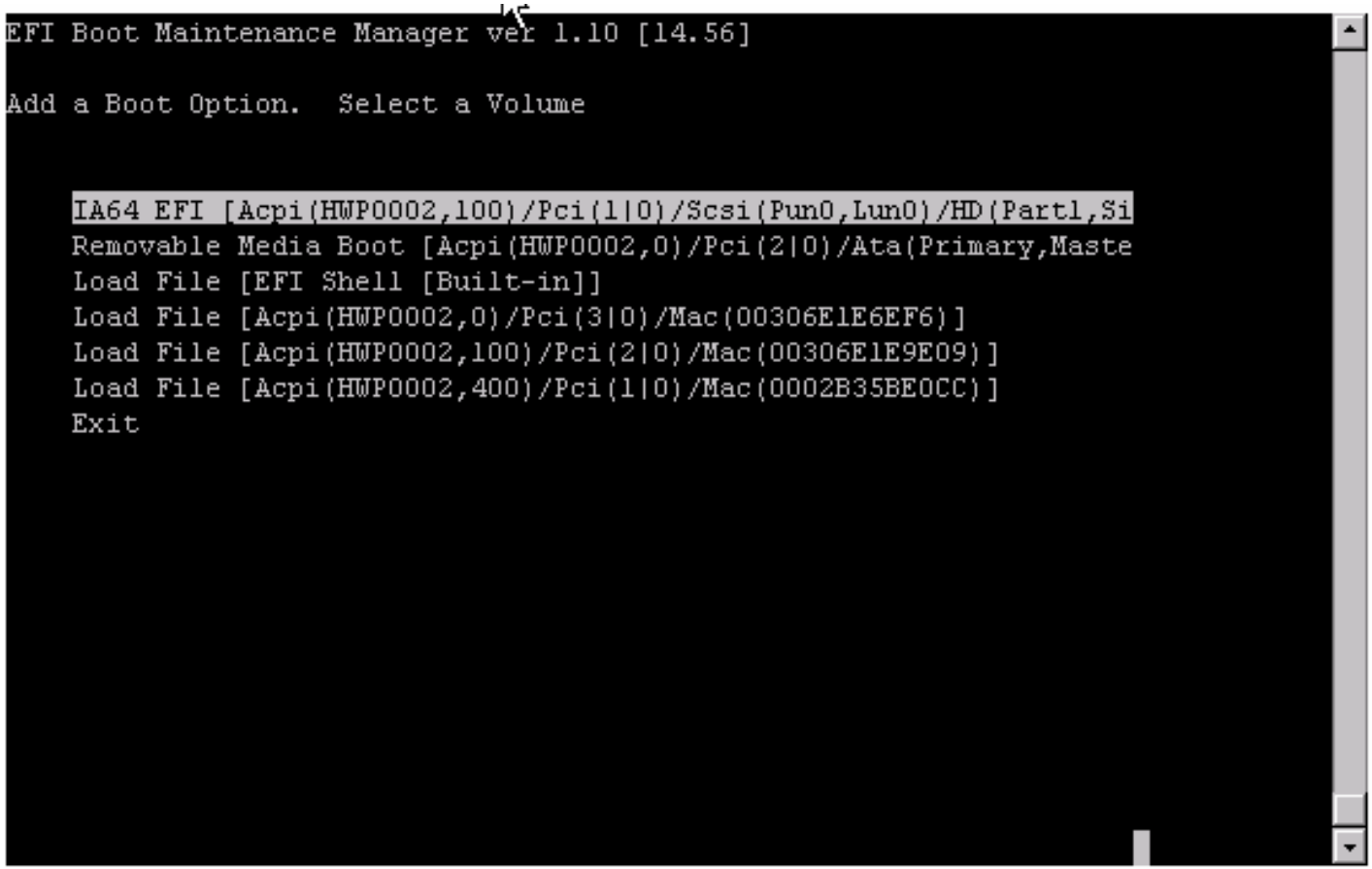
Select ‘Boot option maintenance menu’ from main menu screen and hit Enter

EFI Boot Manager - Adding a boot option



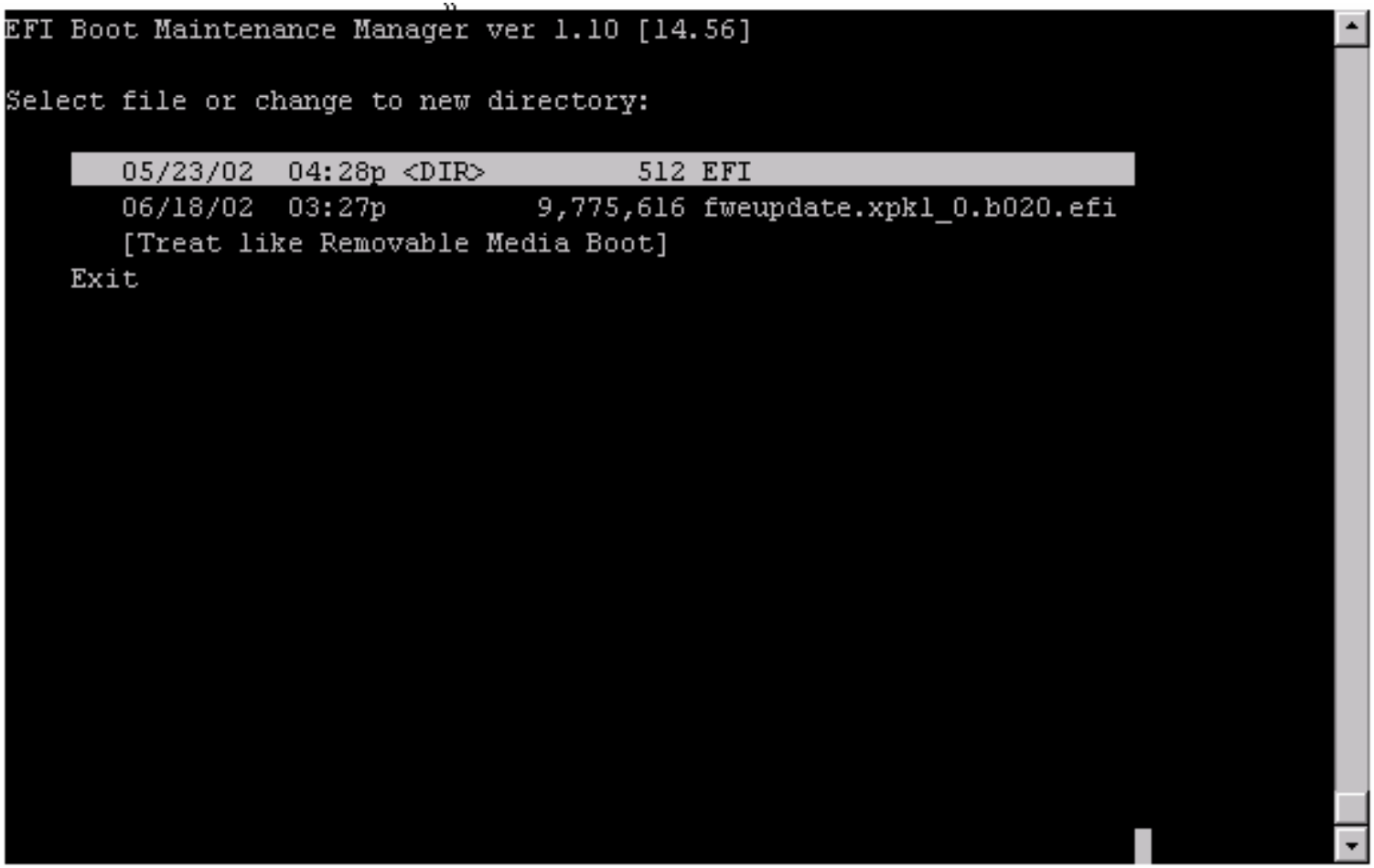
Select ‘Add a Boot Option’ from the EFI Boot Option Maintenance Manager menu and hit Enter

EFI Boot Manager - adding a boot option



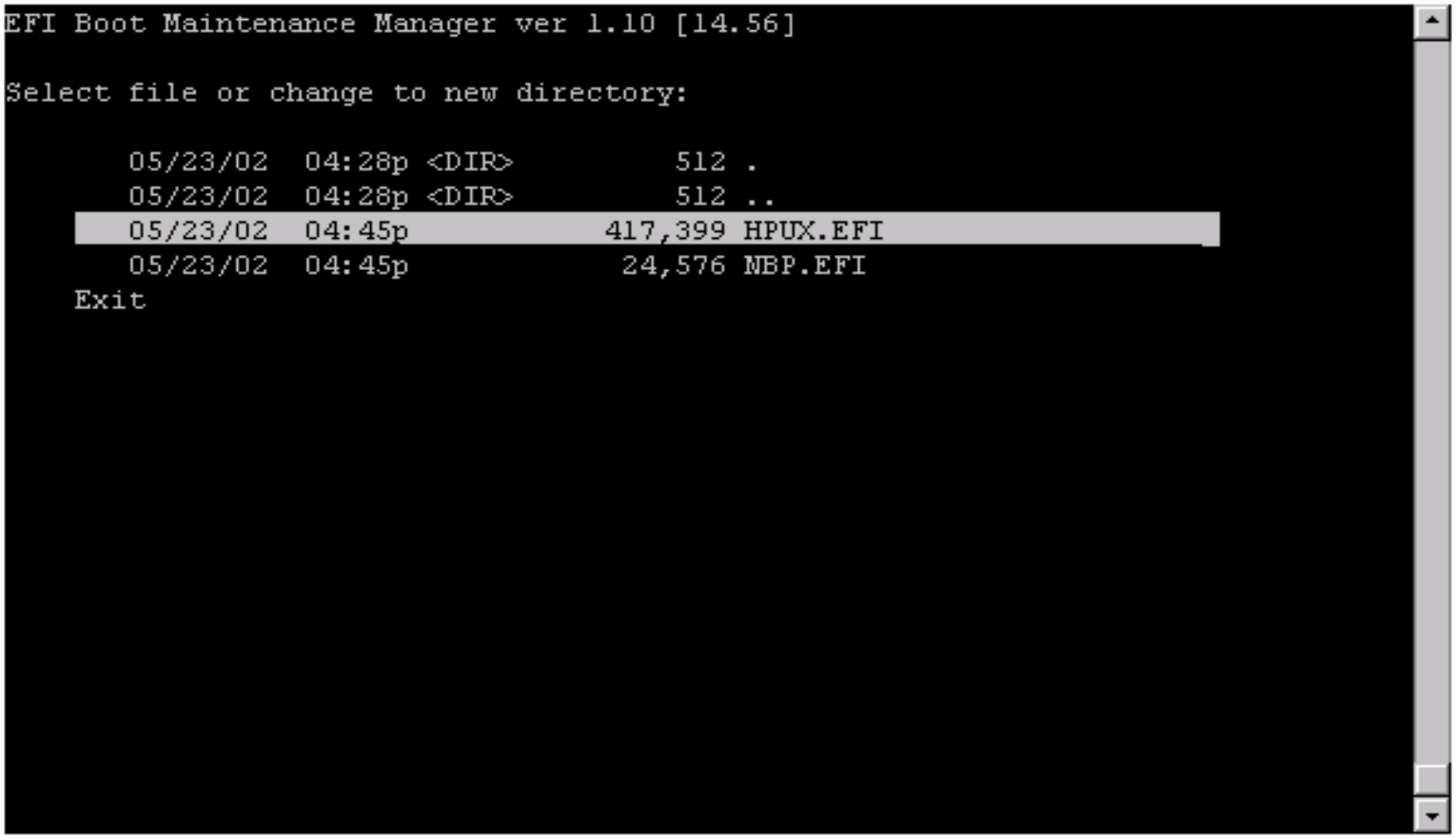
Select the device containing the file from which we want to boot, and hit Enter

EFI Boot Manager – traverse to boot file



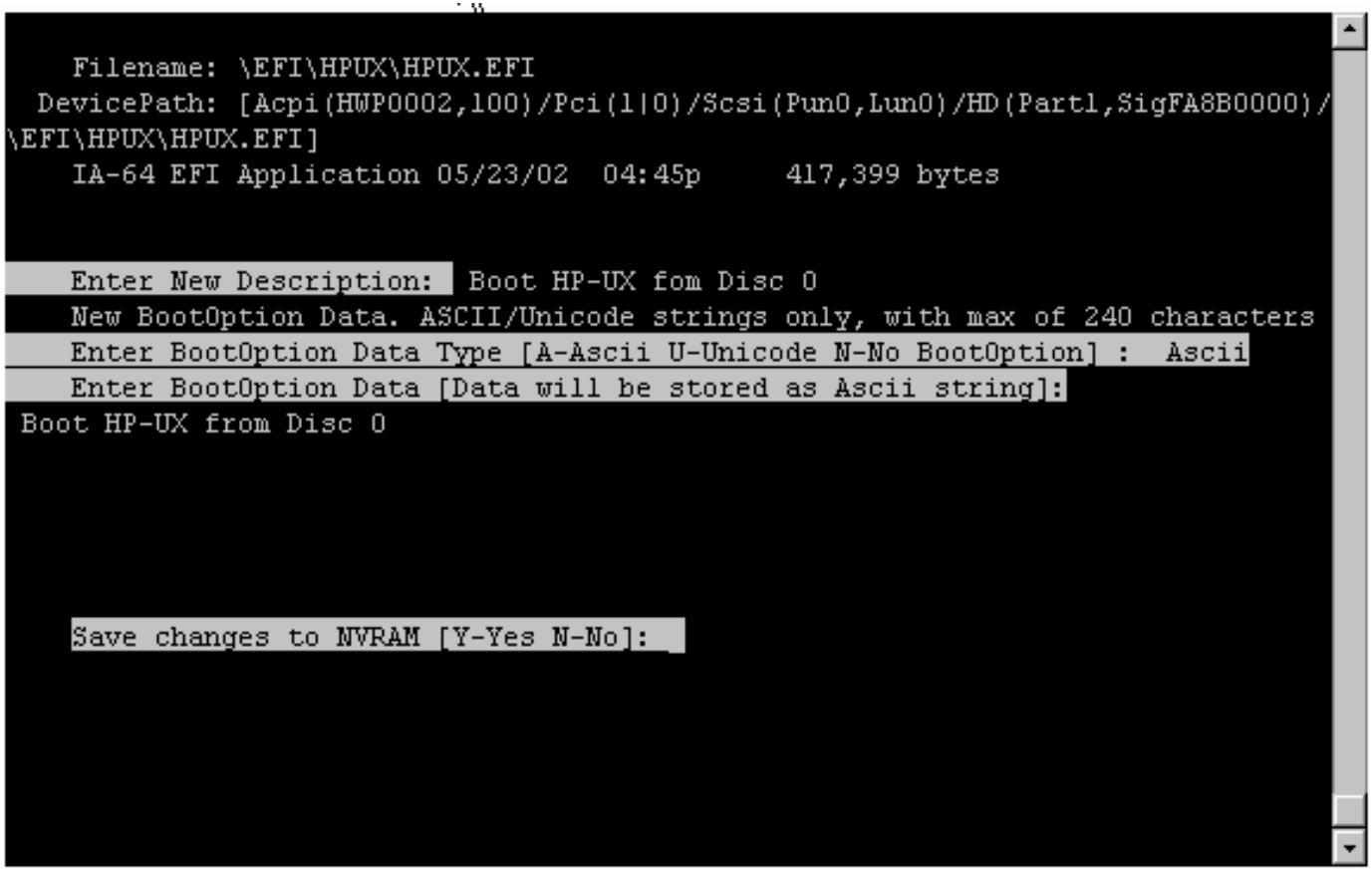
Select directory entries until we reach the desired directory

EFI Boot Manager - adding a boot option



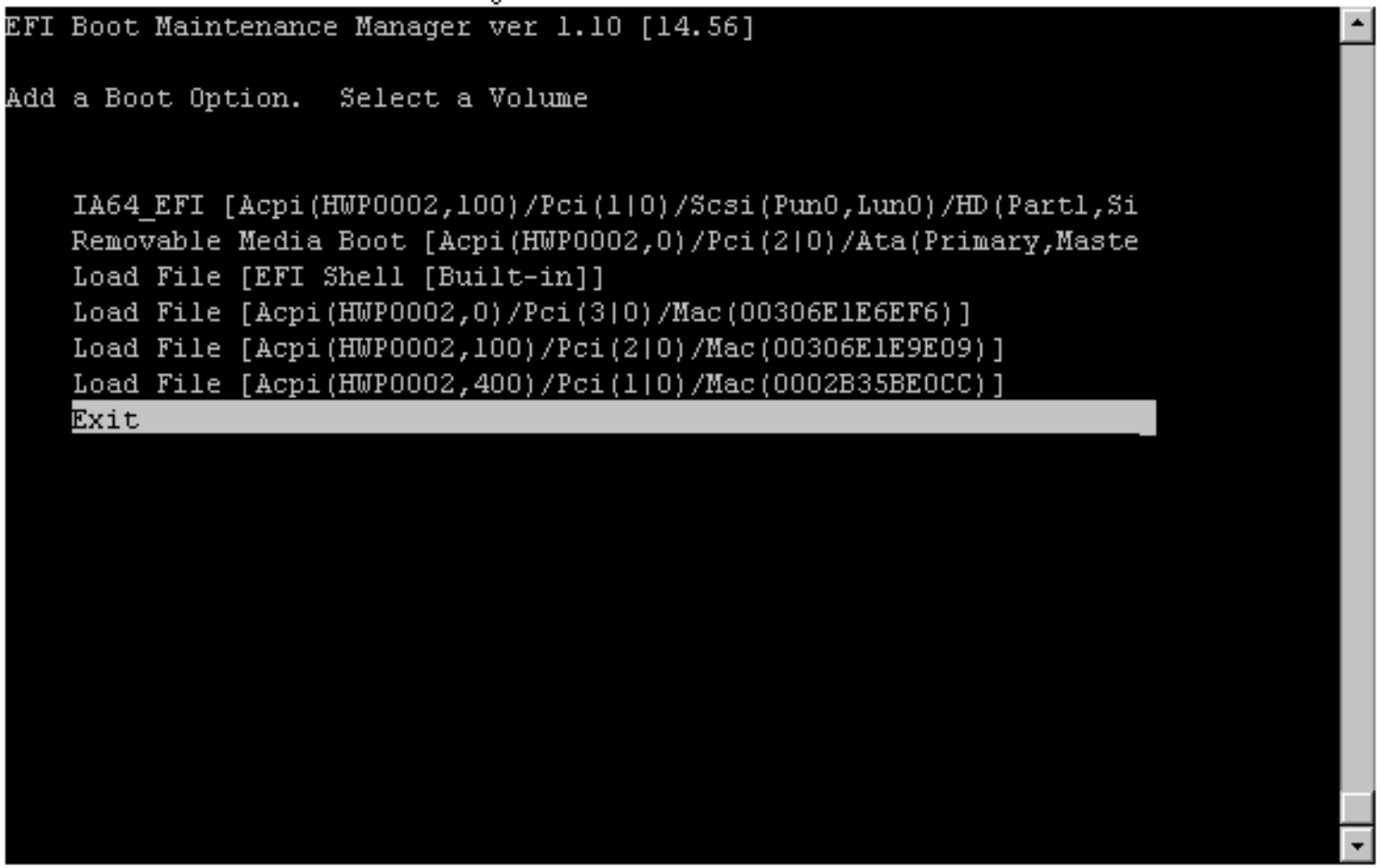
When the desired directory is reached, highlight the boot file and hit Enter

EFI Boot Manager – entering description



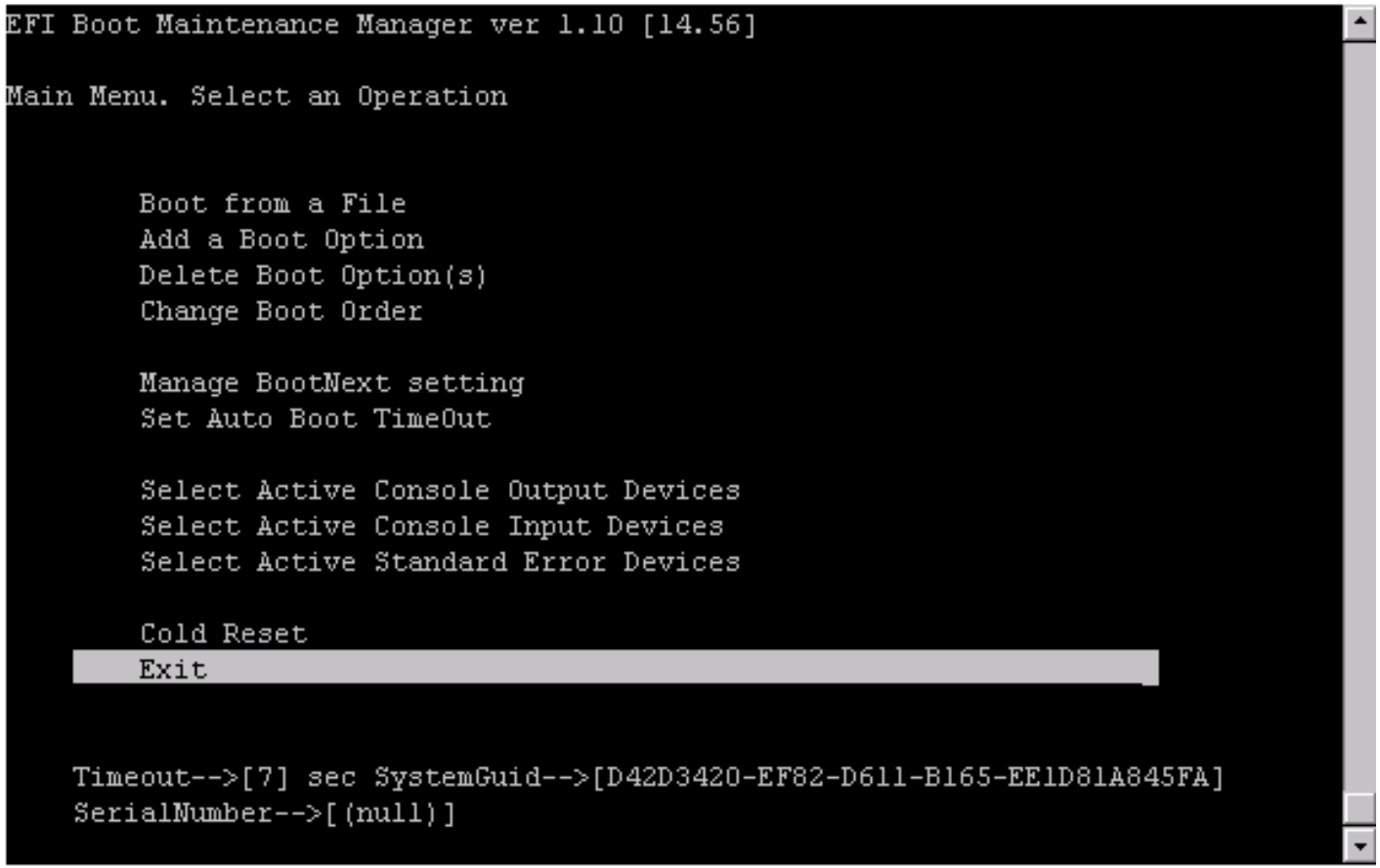
Type in the text for the boot menu entry, the type of data (ASCII or Unicode), and the description, then save to NVRAM

EFI Boot Manager – exit add boot option



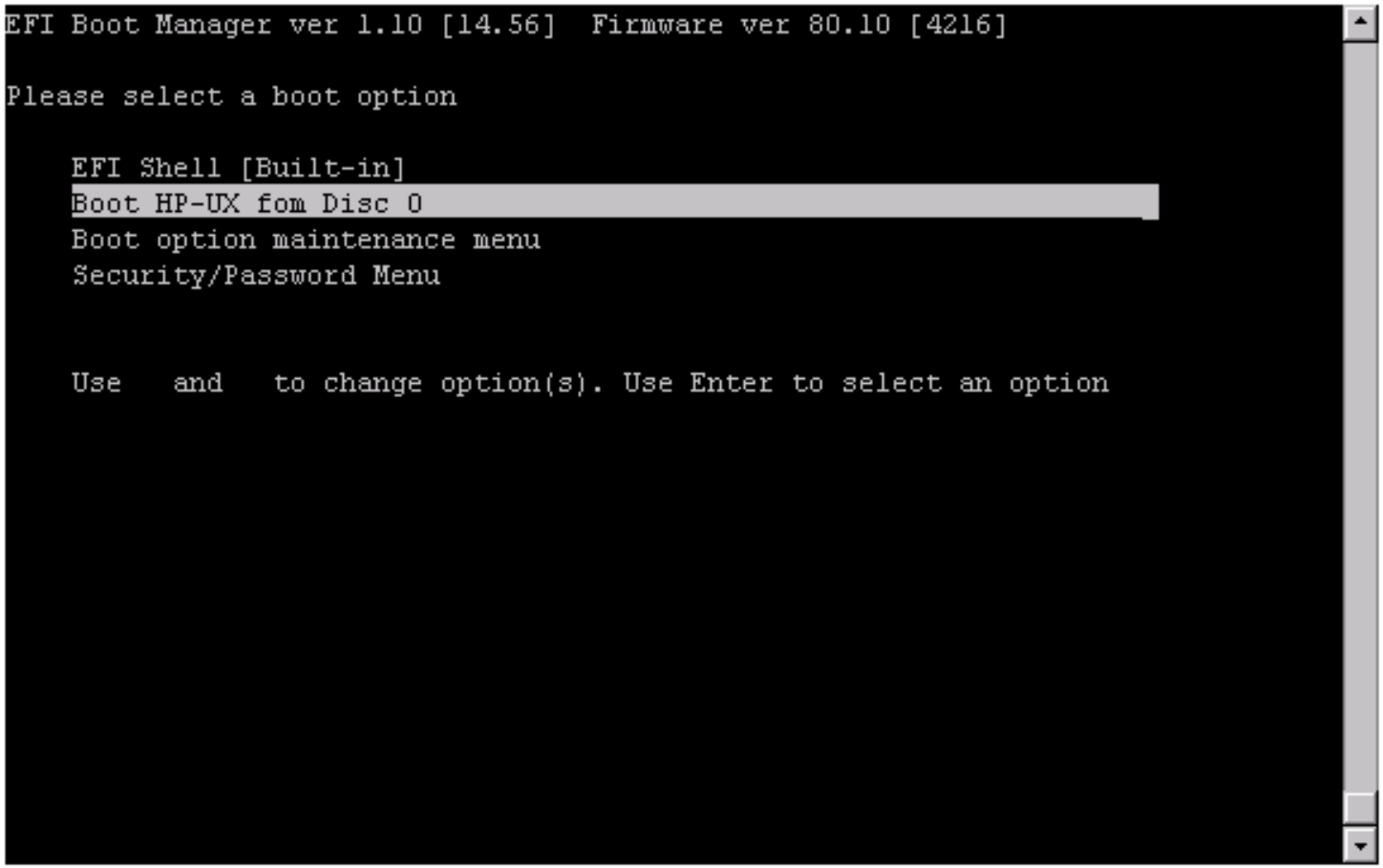
After entering the description and saving it, move down to the ‘Exit’ line and hit Enter

EFI Boot Manager – exit boot maintenance



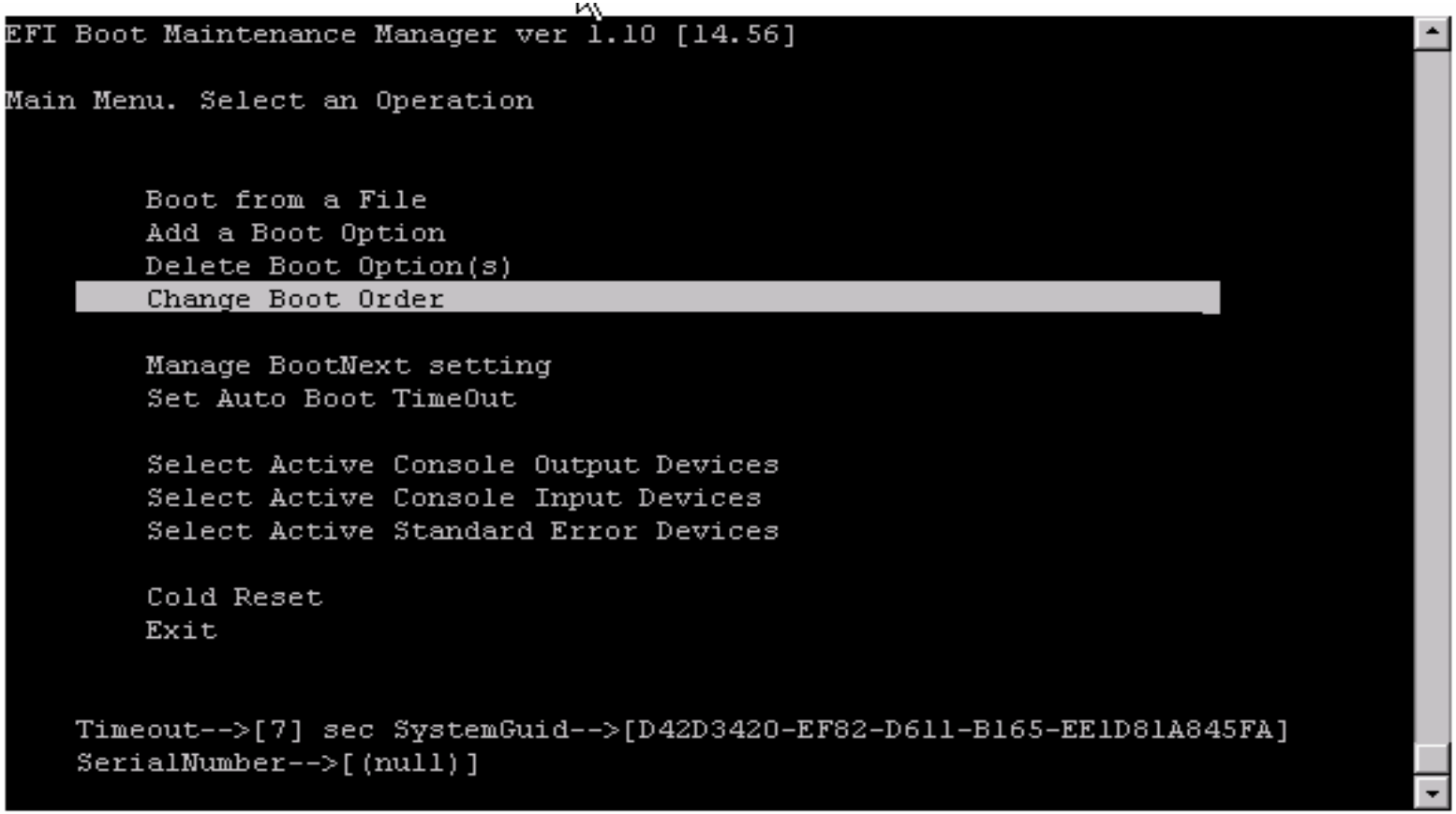
To get back to the main menu, move down to ’Exit’ and hit Enter

EFI Boot Manager – new boot option added



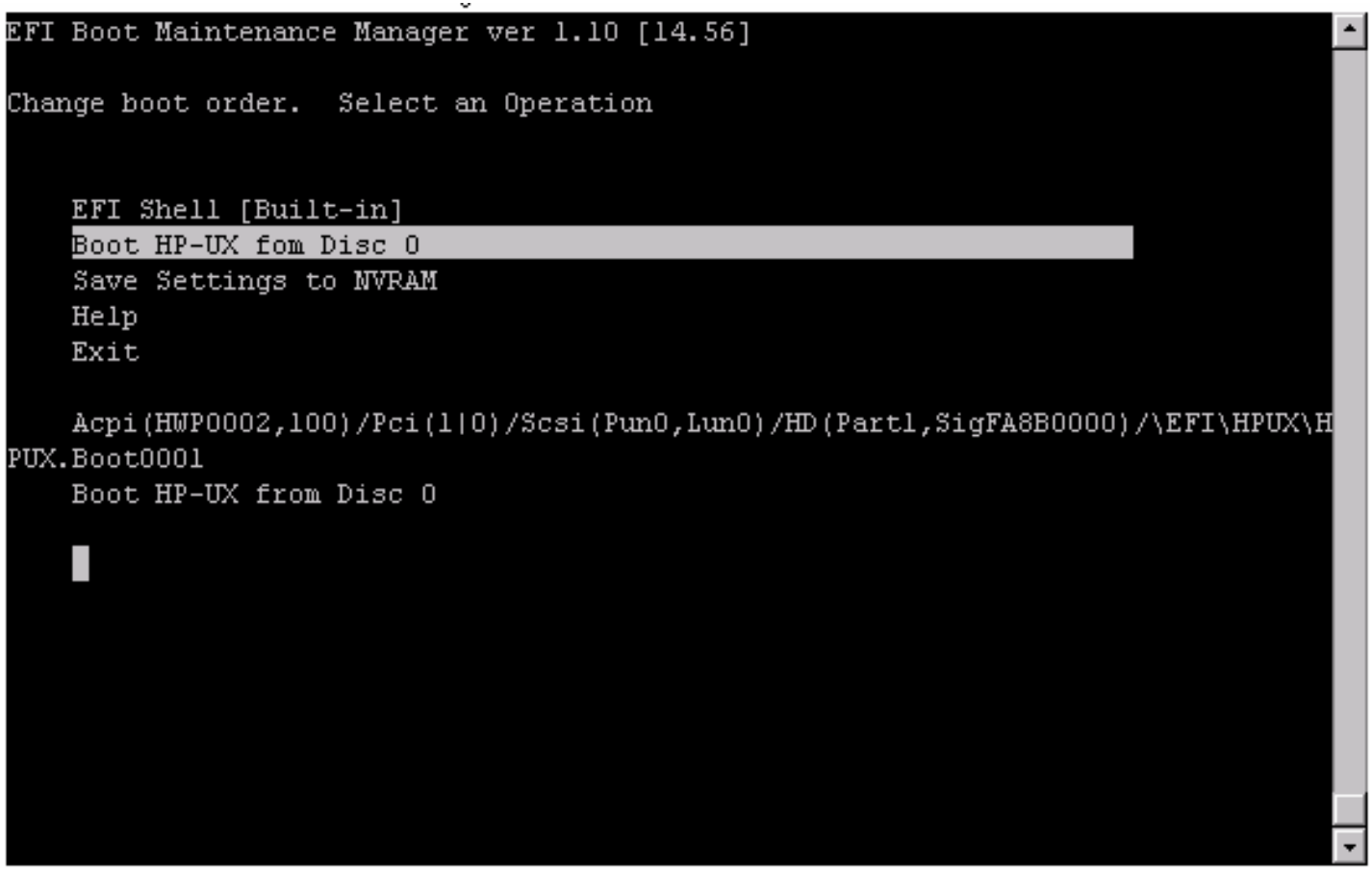
The new boot entry has been added – now we go back to the Boot Maintenance Manager to change boot order

EFI Boot Manager – changing boot order



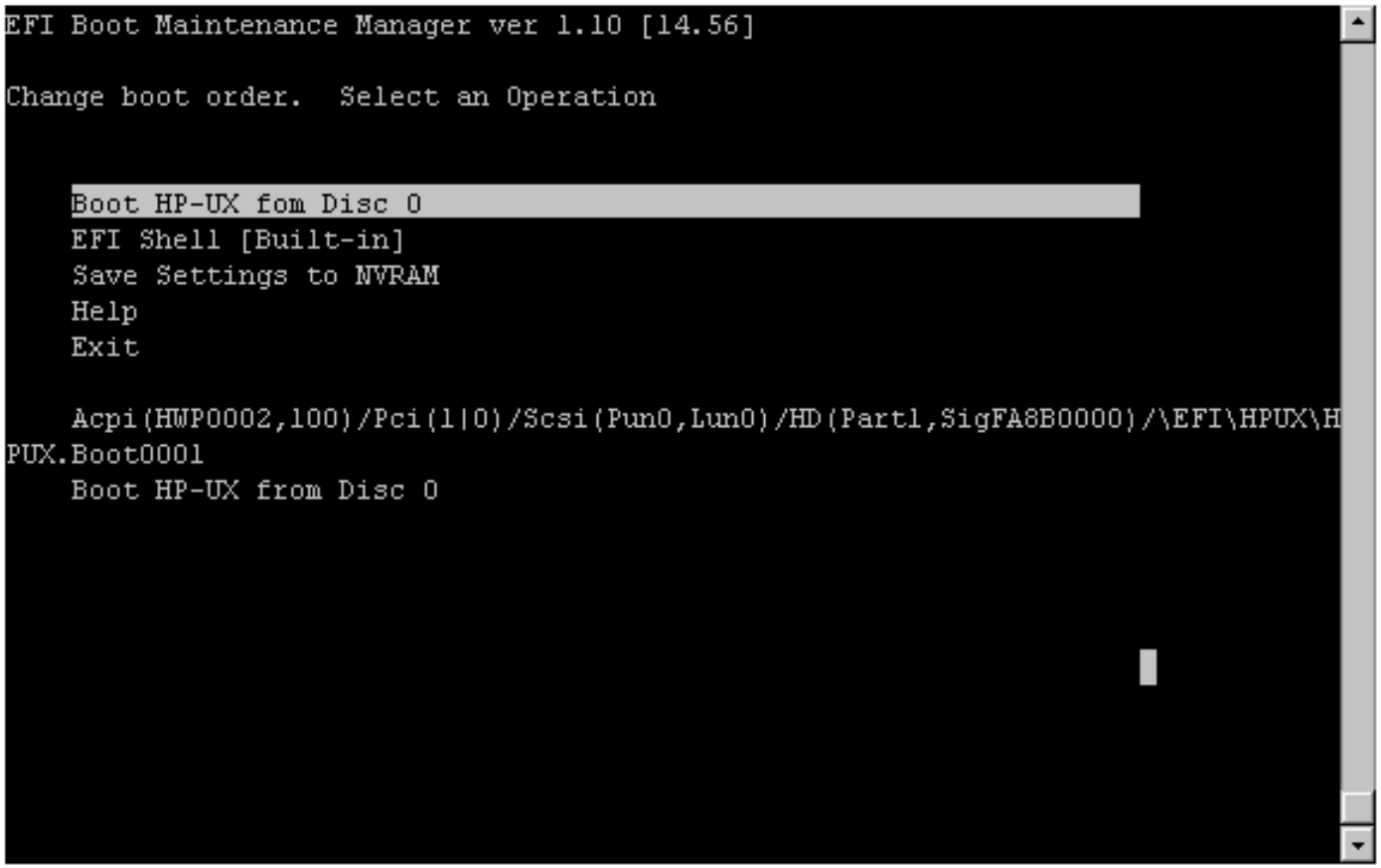
Move down to ‘Change Boot Order’ and hit Enter

EFI Boot Manager – changing boot order



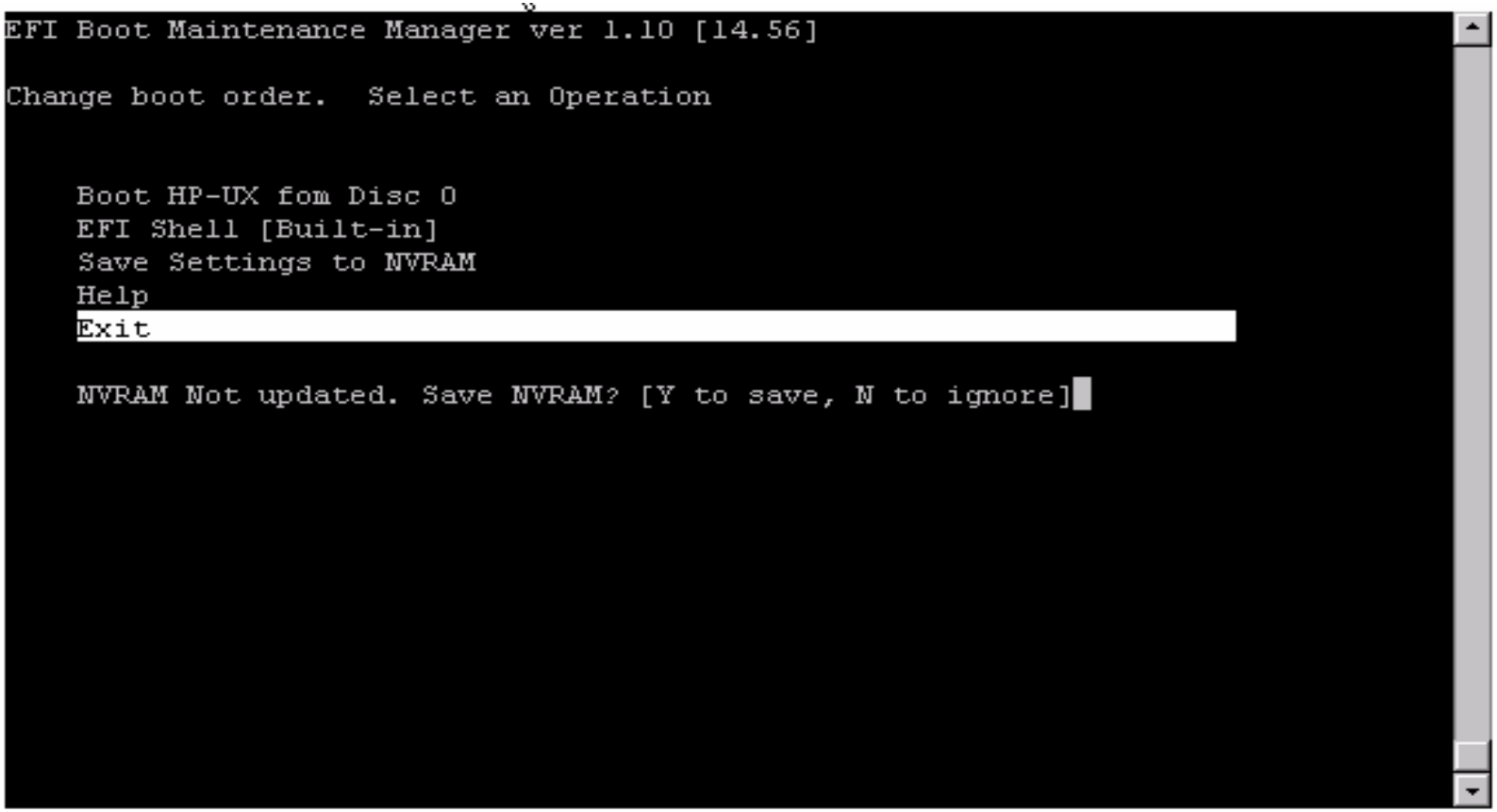
Select the boot menu entry that we want to move up or down in the boot menu

EFI Boot Manager - adding a boot option



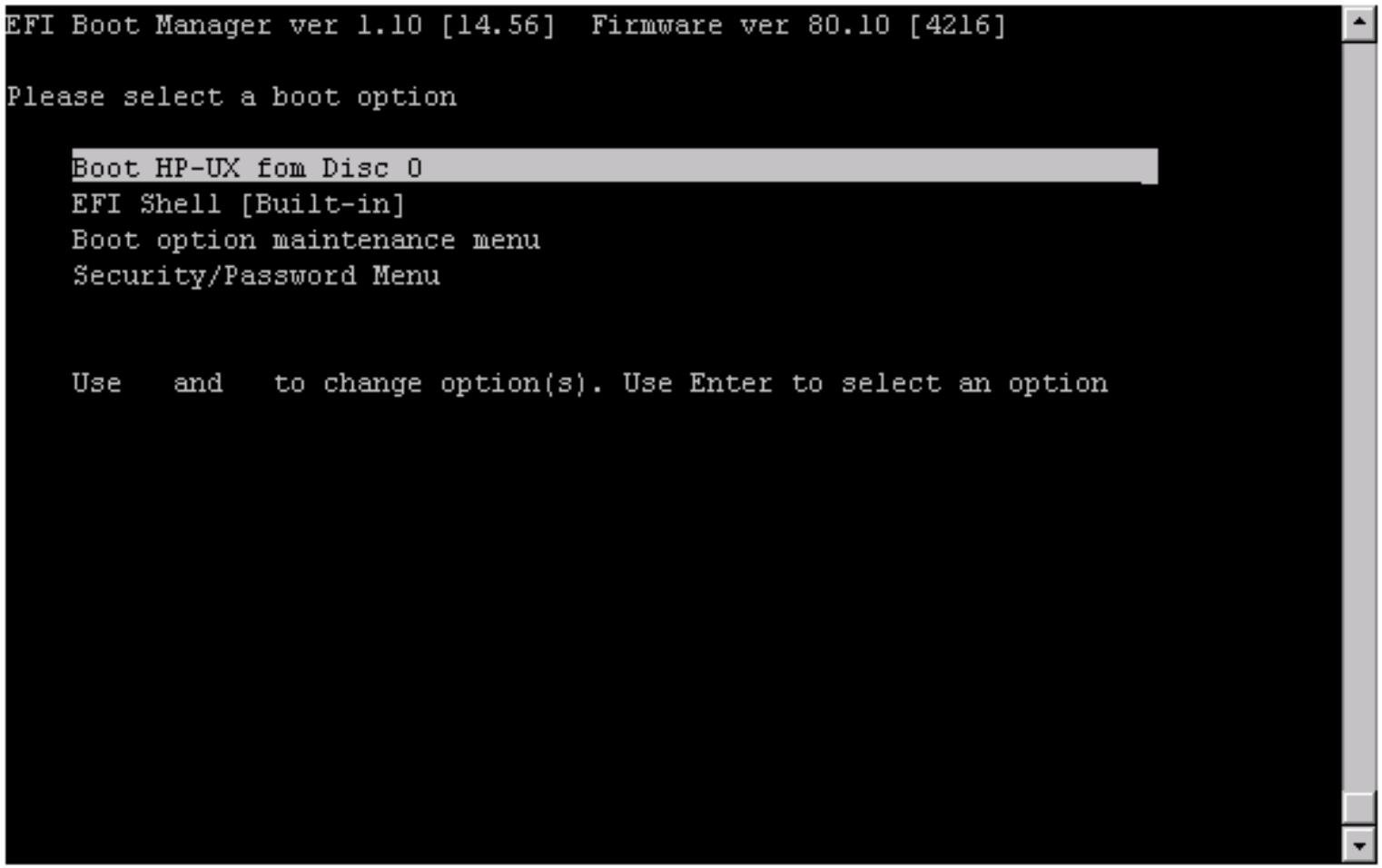
After highlighting the desired entry, use the ‘u’ key to move it up in the menu, or the ‘d’ key to move it down.

EFI Boot Manager - adding a boot option



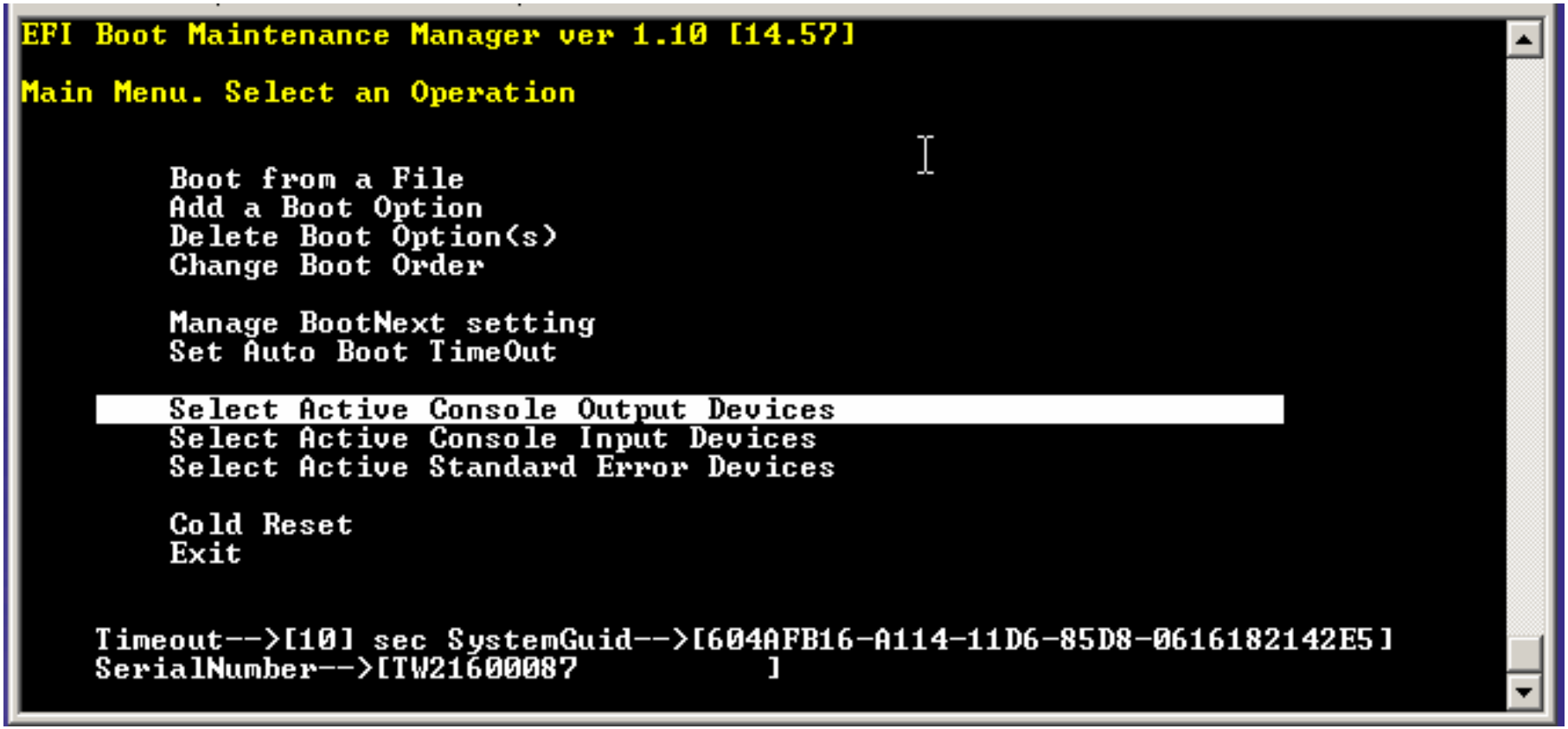
After the entry (or entries) are moved, move down and either save the changes to NVRAM by selecting ‘Save Settings to NVRAM’, or select Exit and save when asked.

EFI Boot Manager – boot order changed



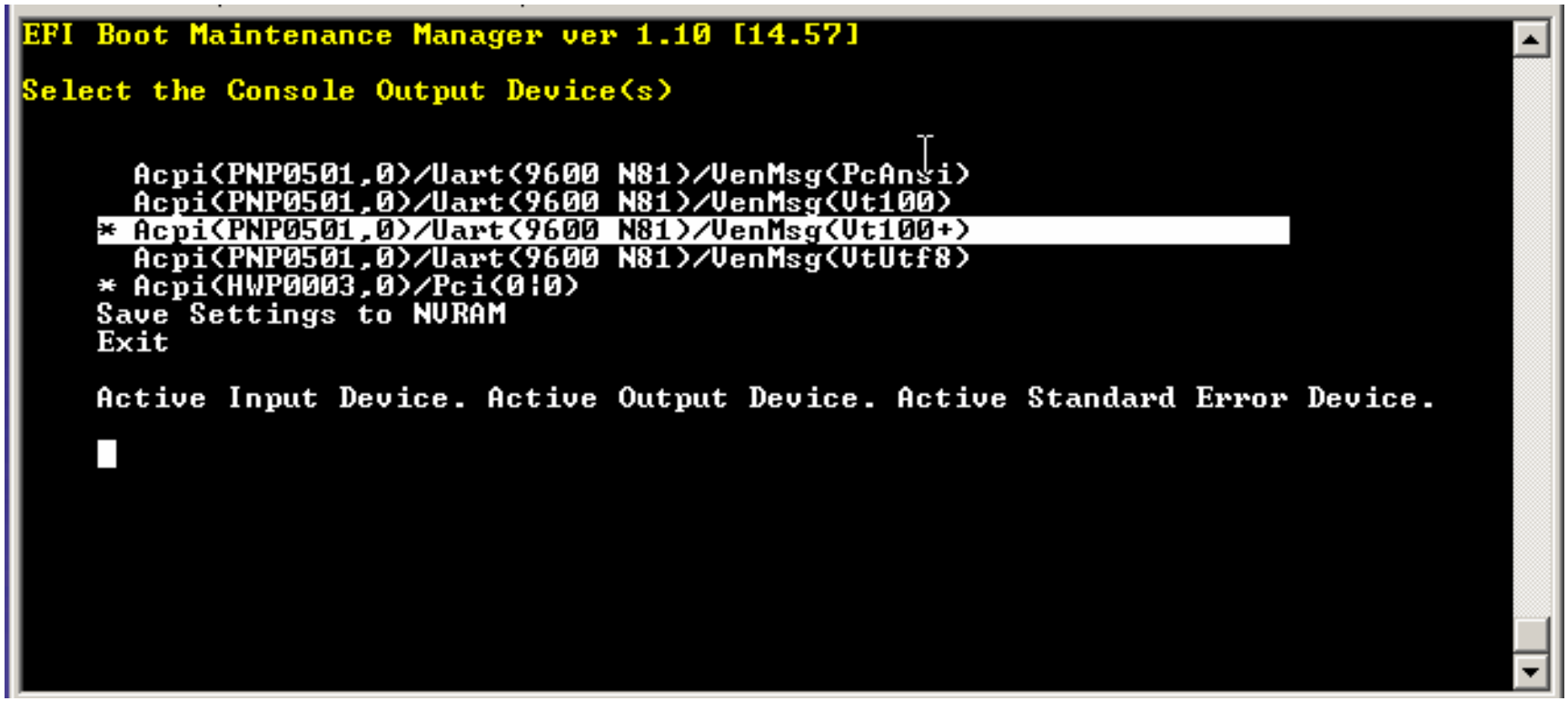
The boot order has been changed. End of this task.

EFI – Console device setting



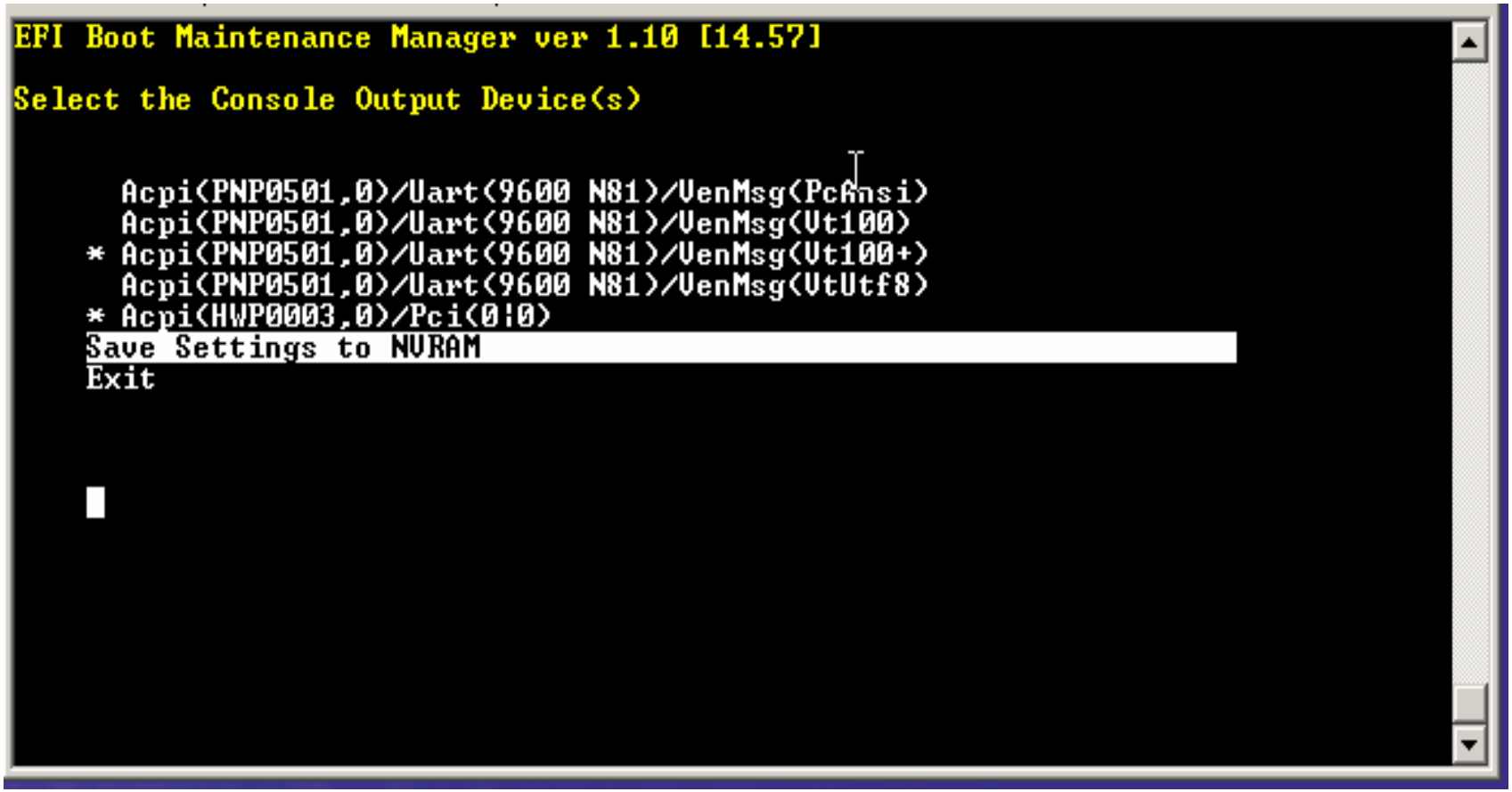
From the Boot Maintenance Manager Menu, select the type of device (Input, Output, or Error) for which you want to change the settings

EFI – Console device setting



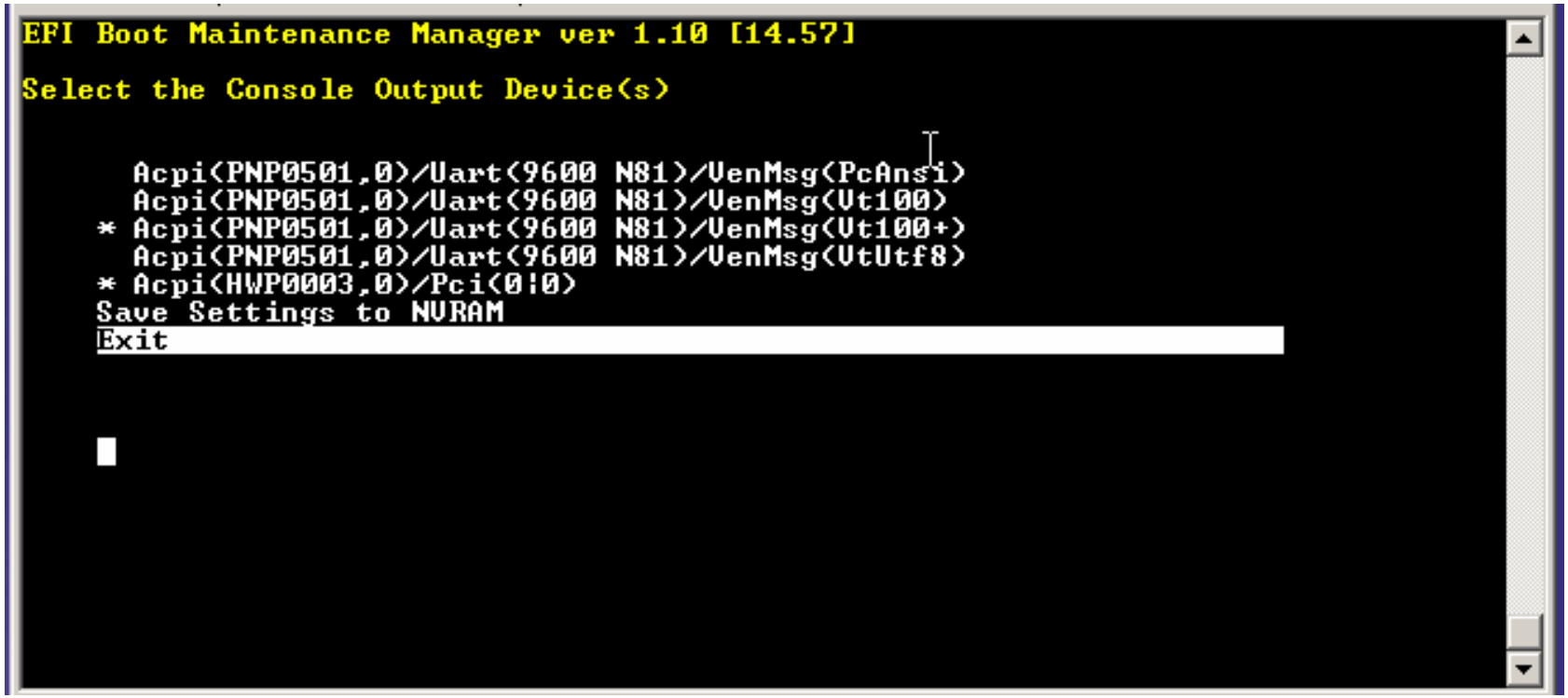
Use the up and down arrow keys to choose the desired device. Devices associated with ‘Uart’ are serial, devices associated with ‘Pci’ are graphics consoles. Press the space bar to enable or disable the device.

EFI - Console device setting



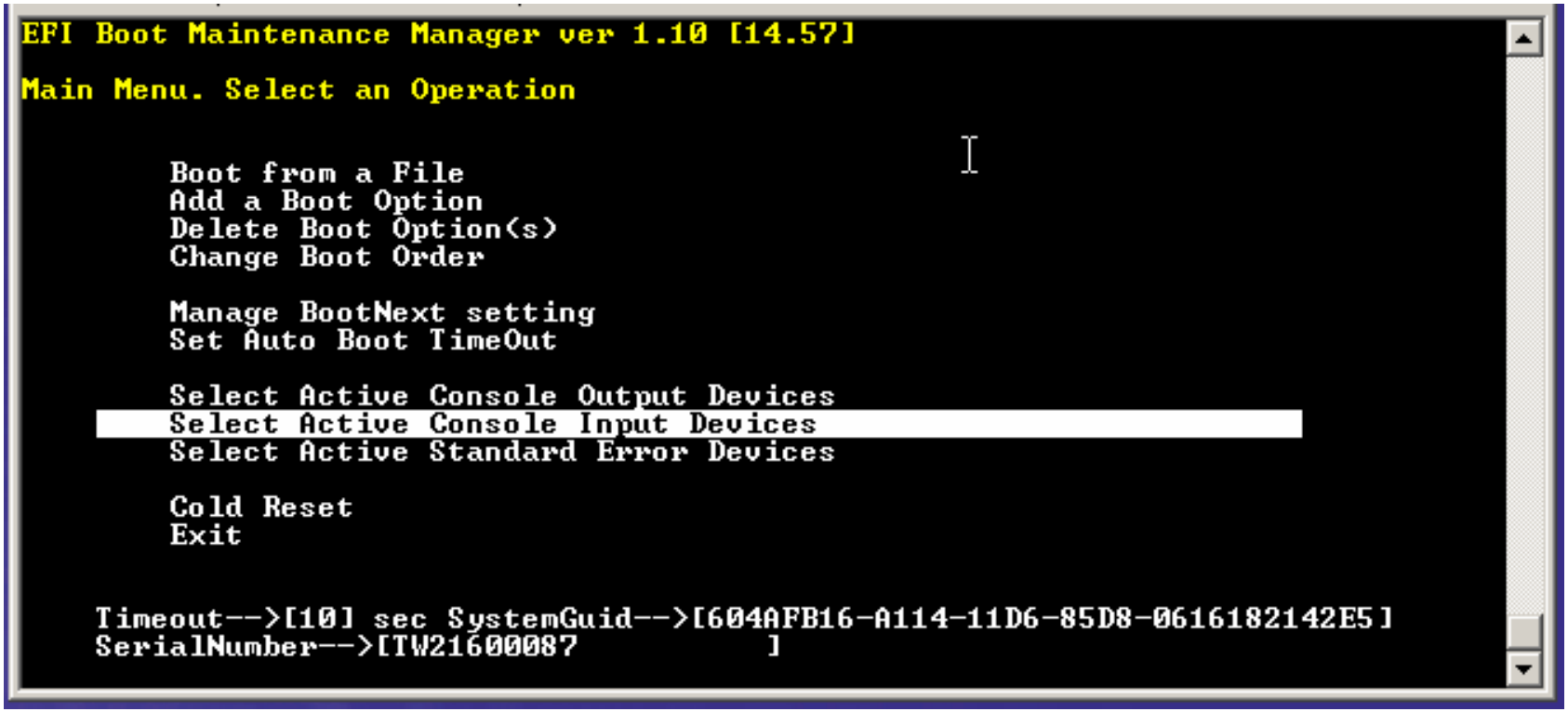
After enabling or disabling the desired devices, move to ‘Save Settings to NVRAM’ and hit Enter

EFI – console device settings



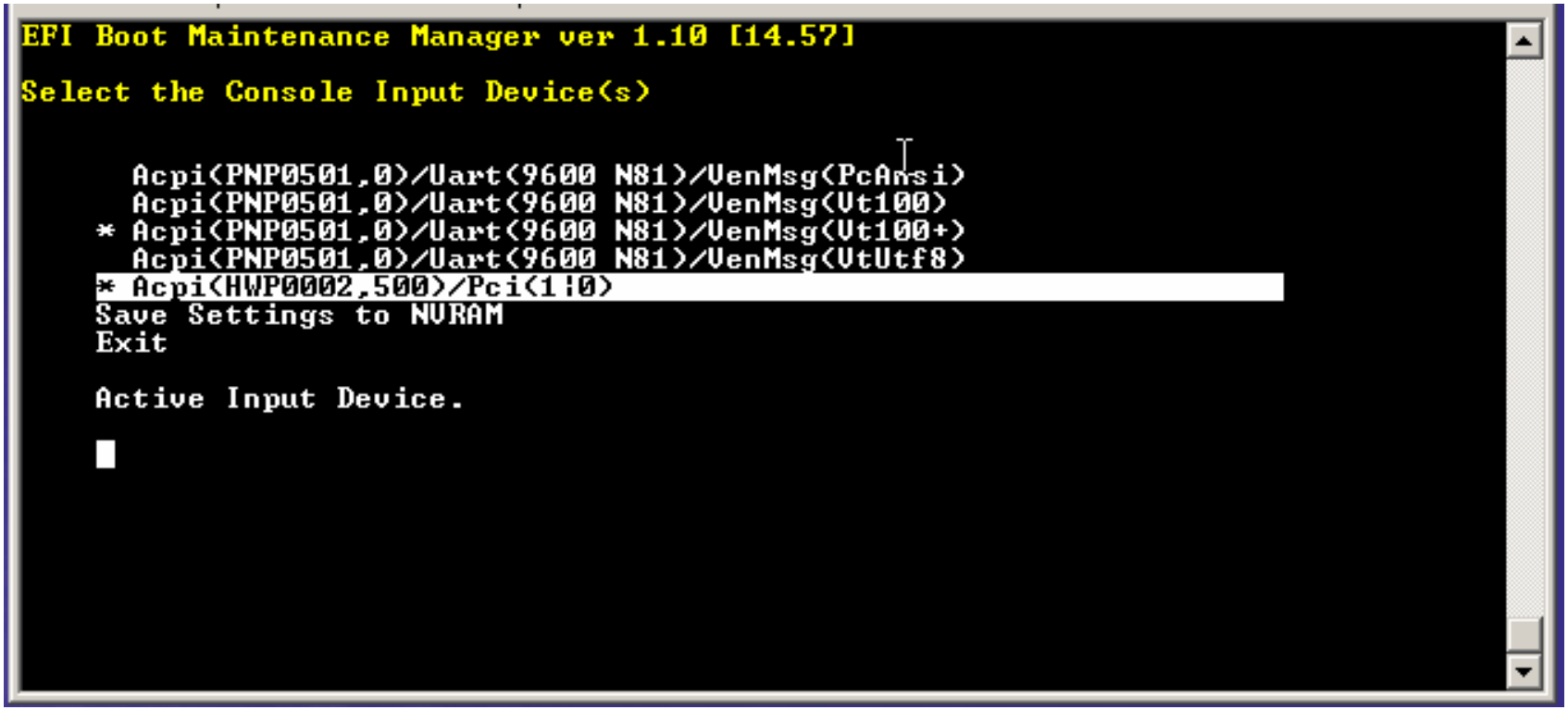
Then move to ‘Exit’ and hit Enter to go back to the Main Boot Manager Menu

EFI – console device setting



Then use the arrow keys to select another device or Exit

EFI – console device setting



The choices are the same for all three types of console devices

Questions?